CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER 70 COMPUTER SYSTEMS
MODELS 72, 73, 74
6000 COMPUTER SYSTEMS

SCOPE REFERENCE MANUAL
MODELS 72, 73, 74 VERSION 3.4
6000 VERSION 3.4

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

REVISION RECORD

REVISION	DESCRIPTION
A	Original printing.
(10-8-71)	
B	Adds Diagnostics (Appendix D) and Glossary. Pages affected are: D-1 thru D-56; Glossary-1 thru
(11-11-71)	Glossary-8. Comment Sheet.
C	Changes and corrections resulting from product development and documentation evaluation.
(7-31-72)	Pages affected are: Front matter; 1-1 thru 1-6; 2-2 thru 2-7, 2-12; 3-3 thru 3-20; 4-1 thru
	4-34, 4-37, 4-38; 5-1 thru 5-22; 6-3, 6-7, 6-9; 7-1, 7-2, 7-5 thru 7-10; 8-1 thru 8-4, 8-8 thru 8-12; 9-5;
	10-1, 10-2, 10-6; 11-1, 11-2, 11-6 thru 11-18, 11-27 thru 11-31; 12-1 thru 12-10, 12-15 thru 12-18,
	12-22, 12-25, 12-26, 12-31, 12-35 thru 12-53, 12-59, 12-62, 12-63, 12-65; Glossary-1 thru Glossary-8;
	A-1 thru A-6; B-1; C-2 thru C-10, C-14; D-1 thru D-72; E-5, E-8 thru E-11; Index 1 thru Index 32;
	Comment Sheet.
D	Changes reflecting the redefinition of the term aphanumeric plus minor corrections. Pages affected are:
(10-20-72)	2-8; 3-1, 3-2; 4-9, 4-11, 4-15, 4-25, 4-27, 4-36; 5-12, 5-14, 5-15; 7-6, 7-10; 8-5, 8-9, 8-13; 9-3; 10-6;
	11-2, 11-5, 11-22, 11-27; 12-3, 12-16, 12-21, 12-22, 12-25; Glossary 3 thru 8; C-1, C-3 thru C-5;
	D-48, D-54; E-10, E-11; Index-9, 22, 28; Comment Sheet.
Publication No. 60307200	

Additional copies of this manual may be obtained from the nearest Control Data Corporation sales office.

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Software Documentation
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

PREFACE

This manual describes the SCOPE 3.4 Operating System for the CONTROL DATA ® CYBER 70/Models 72, 73, and 74 and 6000 Series Computers. It was written for programmers who use all the source languages which operate under SCOPE 3.4, and it includes information of specific interest to those who write in COMPASS assembly language.

Other documents of interest to programmers using SCOPE 3.4:

	Publication Number
SCOPE 3.4 Systems Programmer's Reference Manual	60306500
SCOPE 3.4 Operator's Guide	60327300
Record Manager Reference Manual	60307300
LOADER Reference Manual	60344200
UPDATE Reference Manual	60342500
SCOPE 3.4 Installation Handbook	60307400
SCOPE 3.3 to 3.4 Conversion Aids (SPB)	60358200

SCOPE 3.4 is an extension of SCOPE 3.3. It includes the following new features and changes:

Addition of an integrated scheduler to optimize central memory usage

User libraries creation and maintenance similar to EDITLIB creation of system libraries

Support of SCOPE format records on 9-track tape

New loader with associated restructuring of SCOPE system libraries and overlay/segment structures

Revised REQUEST control statement parameters

Addition of VSN card to support automatic tape assignment to requesting jobs

Extended error processing

Addition of sequential disk pack capabilities for high speed access to a sequential file

Renaming of private disk packs to family disk packs

Additional permanent file functions: ALTER, SETP

Additional parameters for permanent file functions

File action/COMPASS macro change to all skip commands, OPEN, CLOSE, CLOSER

Replacement of SNAP and TRACE debugging aids with TRAP

New extended core storage system

Addition of Record Manager

Addition of File Organizer and Record Manager utilities

Common files are no longer supported

New permanent file utility TRANSPF

Use of non-stop I/O by permanent file utilities

Use of S tapes by permanent file utilities

This product is intended for use only as described in this document. CONTROL DATA cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

APPENDIXES

FIGURES

INTRODUCTION 1

SCOPE consists of a group of programs and subprograms that comprise the operating system for the CONTROL DATA® CYBER 70/Models 72, 73 and 74 and 6000 Series computers. Input, compilation, assembly, loading, execution, and output of all programs submitted to the computer, as well as the allocation of resources for these programs, are monitored and controlled by SCOPE. This file oriented operating system resides primarily on rotating mass storage devices. It uses the versatility and efficiency of the computer hardware to direct multiprogramming. Jobs written in the COMPASS assembly language and a variety of compiler languages, or jobs calling for many different utility systems, can be processed under SCOPE. The product set includes: COMPASS 3.0, FORTRAN 2.3, FORTRAN Extended 4.0, COBOL 4.0, Record Manager 1.0, FORM 1.0, SYMPL 1.0, QUERY/UPDATE 1.0, QUIDDL, SORT/MERGE 4.0, PERT/TIME 2.1, 1700 MSOS Import HS 1.0, 1700 Import HS 1.0, 8231 Import HS 1.0, SIMSCRIPT 3.0, ALGOL 3.0, APT 2.2, SIMULA 1.0, INTERCOM 4.1, BASIC 2.0, MARS VI 2.1, and SAAM 1.0. For compatability with previously compiled binary decks, the relocatable object-time libraries for FORTRAN Extended 3.0, COBOL and SORT/MERGE are also included.

HARDWARE OVERVIEW

The minimum hardware requirements consist of: the computer (including 50,176 words of central memory storage); any combination of disks, disk packs, or drums to provide 24 million characters of mass storage; a card reader, card punch, and printer (with controllers); and two magnetic tape units. Larger systems can be obtained by including optional equipment such as: additional central memory, extended core storage (ECS), additional card readers, punches, printers, and tape units at the central site, as well as remote terminals.

The computer is composed of central memory, one or two highspeed central processors, seven to 20 peripheral processors, and a display console. Up to 15 system or user jobs can be in central memory simultaneously, sharing the central processor registers at scheduled intervals. Each job in central memory gains access to the central processor alternately with the other jobs until execution is complete.

The peripheral processors are used to input jobs to the computer, move jobs between central memory and mass storage, transfer input or output data between mass storage or peripheral devices and jobs in central memory, and dispose of output from completed jobs. They relieve the central processor of all input/output tasks, enabling it to devote more time to program execution. Each peripheral processor contains its own memory and is actually an individual computer that operates independently of the other processors.

The display console includes two CRT screens which display information about the system and the jobs in process. The console also includes a keyboard through which the operator can enter requests to modify stored programs and display information about jobs in or awaiting execution.

Further information about the computer hardware can be found in the CDC CYBER 70/6000 Series Computer Systems Reference Manual. Details about the movement of information between the central processor registers and central memory are presented in that manual and also in the reference manual for COMPASS.

MAIN CONCEPTS OF SCOPE

All SCOPE routines are recorded in a file on a mass storage disk or drum. Copies of routines used most frequently also reside permanently in central memory or in ECS so they can be executed with minimal delay. Others are called into memory from ECS or mass storage only when needed to ensure that a maximum amount of memory remains free for user jobs.

The portion of SCOPE residing in central memory consists of system tables and pointer words used for communication between SCOPE routines. Also, some frequently used routines that can be called into peripheral processor memory on a transient basis are stored in central memory or ECS because they can be loaded from that area much faster than from mass storage. Central memory and ECS areas reserved for SCOPE cannot be overwritten by user jobs.

One peripheral processor (PP0) holds only the Monitor routine, MTR, that oversees and controls all SCOPE functions. Part of Monitor also resides in central memory. Monitor functions include allocating hardware to user jobs, as well as coordinating the activities of all other PP's.

Another peripheral processor (PP1) is devoted exclusively to routine DSD that drives the system display console and input keyboard. This routine interprets and processes all requests typed by the operator and displays all messages from SCOPE to the operator.

Each remaining pool PP contains a resident routine that services requests from Monitor, loads and executes programs as assigned by Monitor, and provides a communications interface between Monitor and the program loaded. The programs which peripheral processors load and execute include routines for handling input/output activities associated with all phases of job execution, such as: reading a job from a card reader onto mass storage or moving a job between central memory and mass storage, sending output to mass storage or printer, or reading and writing information on magnetic tapes.

Each resident routine in a pool PP contains pointer words that refer to a communication area in central memory. The PP resident routine contains a subroutine that uses these pointers to continually examine the communication area for requests from Monitor.

BASIC SCOPE FUNCTIONS

When a job enters the computer, SCOPE processes the job, assigns the hardware resources required, and ensures that input, output, and system files needed by the job are made available to it. These three functions are interrelated and mutually dependent; resource and file management take place through the entire course of job processing.

Job processing consists of the following tasks: loading the job into the computer, assigning it system resources (central memory storage, magnetic tapes, etc.), executing the job, terminating the job, and processing its output.

When the user's program and associated data is in card form, the job consists of these cards preceded by a group of control cards. Control cards request such functions as assembly or compilation, loading, and execution. They also request equipment or files required by the job, and the amount of time and central memory storage or ECS it will need. In addition, processing priority with respect to other jobs and special comments to the computer operator can be specified on these cards.

When the user's programs reside on mass storage or on tape, the job may consist only of control cards that direct the loading and execution of these programs and ensure that the hardware and files needed for their execution are available.

At the central site, job processing is initiated by loading the job into the card reader. SCOPE takes control and assigns a PP that transfers the job from the card reader to mass storage. The job is then in the input queue, awating availability of system resources. When they are available and the priority of the job permits, SCOPE designates another PP to read the job into central memory. The job begins execution, sharing access to the central processor with other jobs in execution. During execution, peripheral processors are assigned to handle any additional input or output files and associated hardware. If requested by control cards, SCOPE assigns additional files or hardware to the job, including tape or disk files or ECS.

Jobs in central memory share the central processor. The job using the central processor may relinquish control when it must await completion of input/output. or Monitor may interrupt the job and give control to another job.

When a job is completed or when it is terminated because of an error, SCOPE frees the central memory areas and releases the files assigned to the job. The output from the job is then in the output queue. When the peripheral devices requested for the output are available, SCOPE writes from the output queue onto these devices. Typical output devices are the printer and card punch. Following the output from the job itself, the dayfile containing chronological information, diagnostic messages, and accounting data about the job is output.

MULTIPROGRAMMING

Multiprogramming means that more than one job can be in process in central memory at one time. At any given instant, only one job can be using the central processor (or each central processor in a dual CP configuration). However, several can be performing input/output. In fact, a job can have more than one input or output operation in progress simultaneously.

CENTRAL MEMORY USAGE

Each job in process occupies a contiguous block of words in central memory. References to all addresses within each block are made in relation to the reference address (RA) which is the first address in the block. The length of the block is the field length (FL) of the job. If a user's job references a location outside its field length, the hardware senses this error; the reference is not carried out, thereby protecting all other jobs and system programs in central memory from being accidentally overwritten. For this reason, the user can consider his job as a program running alone in a computer with a central memory the size of his field length.

Two distinct areas of central memory are reserved for portions of SCOPE; they cannot be addressed by user jobs during execution. The low core area (the beginning addresses of central memory) contains the central memory resident routines of SCOPE, the systems tables, pointer words, communication areas used to link the peripheral processor and central memories, and subroutines used often by both central memory and the peripheral processors. The high core area comprises the last (highest numbered) addresses in central memory. It holds information about files on mass storage and is referenced during data transfer between such files. The relationship of low and high core to the remainder of central memory is shown in figure 1-1. As shown, the first address is at the extreme low end of central memory and the last address is at the extreme upper end.

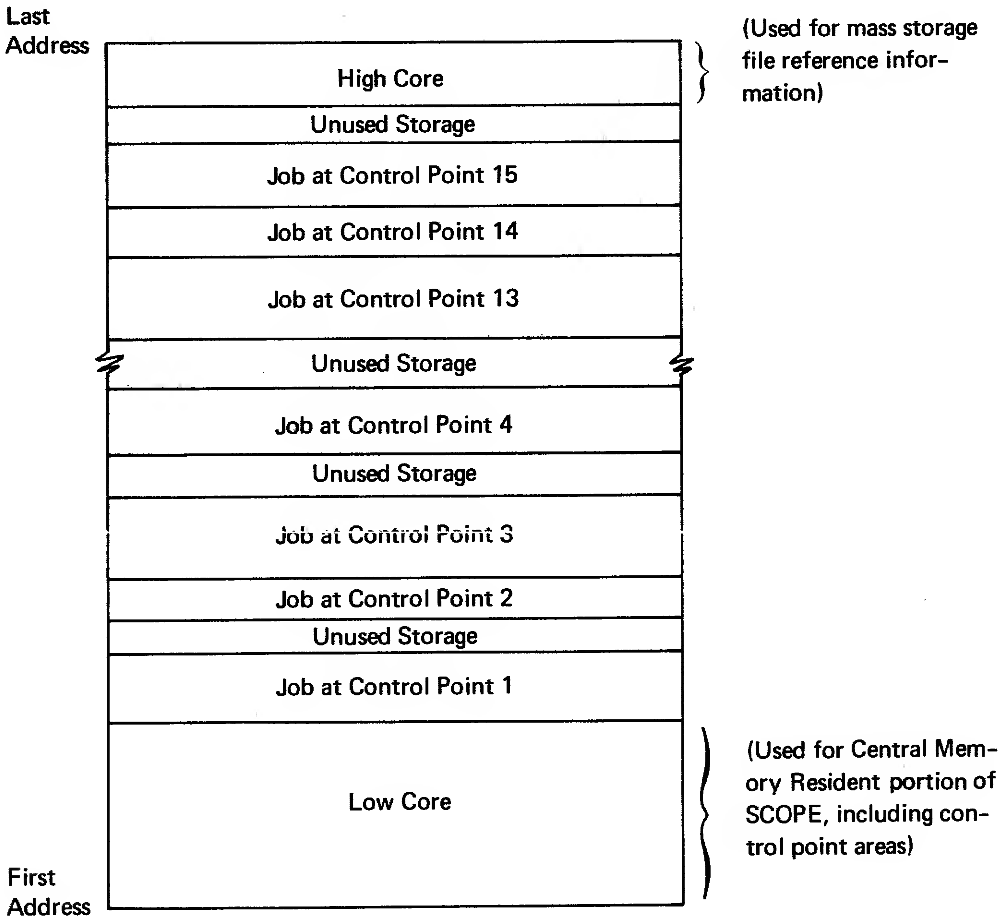


Figure 1-1. Central Memory Allocation

EXTENDED CORE STORAGE USAGE

Extended Core Storage (ECS) is a mass storage device similar to central memory. The large amount of storage and the very fast transfer rates of ECS make it ideal for use as a buffer between central memory and slower peripheral devices, a highspeed swapping device, and storage of large data arrays, as well as a residence for system programs.

CONTROL POINTS

Every job in central memory is related to a SCOPE control point. Each control point interrelates the following elements common to a particular job: The central memory field length allotted; other hardware and files used by the job; and a block in low core, called the control point area, that contains reference information about the job. The control point area contains such information as the job name, processing time accumulated, related control statements, and the job's exchange jump package.

Up to 15 control points (numbered 1 to 17 octal) are available; therefore, up to 15 system or user jobs may be active at control points simultaneously. Control point 0 is used to identify all hardware and software resources not presently allocated to user jobs or those used only by SCOPE.

When jobs are input from a card reader or when job output is transferred to the printer or card punch, the SCOPE input/output routine (JANUS) that handles these functions must reside at one of the control points.

CENTRAL MEMORY ALLOCATION

The position of central memory storage allocated to each job is related to the control point number to which the job is assigned. This assignment is made and maintained in numerical order. Thus, the job at control point 2 will always follow that at control point 1 in memory, and the job at control point 3 will follow the job at control point 2, as shown in figure 1-1.

Through a dynamic relocation process, jobs are moved up and down in storage to make room for new jobs assigned to control points. This process takes place continuously as central memory is required or released. For example, jobs may be running at all control points except control point 2 and a new job is assigned to control point 2. If sufficient contiguous storage is not available for the new job, SCOPE will relocate other jobs as necessary to provide sufficient contiguous storage. Each job will be moved as a block, and only its reference address (RA) will be changed accordingly within the appropriate SCOPE reference tables. It might be necessary to relocate the jobs at both control points 1 and 3, or only one of them. If the job at control point 3 is relocated, it also may be necessary to move one or more of the jobs following it; but the order of the jobs within central memory remains the same. When a job is moved in storage, Monitor suspends all user program activity at the control point, waits for all PPs assigned to the control point to clear their field access flags, and then starts the SCOPE routine that moves the job. When the move is complete, the RA of the job is modified and job activity is resumed.

JOB PRIORITIES

The user can request one of several priority urgencies for his job or leave the assignment to SCOPE. Periodically, SCOPE adds a time increment to compute the job priority used in allocating and deallocating system resources. This time increment reflects the time consumed in waiting for resource availability, ensuring that no job will wait indefinitely for a resource.

JOB SWAPPING

If a job with high priority enters the system, existing jobs of lower priority may be swapped out or rolled out.

When a job is swapped out, all information reflecting the current status of the job is written to a mass storage file. The field length and control point associated with the job are made available to the Scheduler. As control points and central memory become available, swapped out jobs are swapped back in to continue processing. A job can be swapped into any free control point; thus, a job may run at several different control points before it reaches termination.

If a job is currently assigned ECS or non-allocatable equipment (magnetic tape, family packs or sequential packs), job swapping is not possible; however, such a job can be rolled out to obtain additional central memory. The job's field length is written to a rollout file before the field length is freed for another job. The control point is not released when rollout occurs.

If a job is waiting for a permanent file to become available or for a magnetic tape or other equipment to be assigned, the job can be swapped or rolled out automatically. When the file or device becomes available, the job becomes eligible to be swapped or rolled back in.

Swapping or rolling may increase the total time that a job spends in the computer, but it will not affect the amount of central processor time used by a given job; and it should help overall processing. Job swapping and job rollout are controlled by the Scheduler. The most important system effect is to maintain high central processor utilization. Frequent short central processor access is balanced with longer, less urgent, access.

EXCHANGE JUMPS

A program gains or relinquishes the central processor through an exchange jump instruction. When this instruction is executed, the program using the central processor is interrupted. Procedures that Monitor uses to make the exchange depend on the hardware available on each system.

The control point area, which resides in low core, contains a 16-word exchange package which contains the information used directly in exchange jumps: the most recent contents of all central processor registers, the RA and FL in central memory and ECS, and the address of the next instruction to be executed (the program address). This package appears as part of the output from jobs that terminate abnormally, and it can be requested by a dump control card.

JOB/SCOPE COMMUNICATION

Since no instructions within a job can directly access memory locations outside its field length, an area must be reserved within the job's FL that SCOPE checks periodically to maintain communication with the job. This area is composed of the first 100(octal) locations in the job's FL, location RA+0 through RA+77. The user program actually begins at location RA+100. RA+0 is reserved for hardware use (and for the pause flag and sense switches); if an arithmetic error occurs, the type of error and the location of the error are stored in RA+0. Requests from the job to SCOPE are stored in RA+1. They can be requests for input/output, a signal to SCOPE that processing is complete, or a request to terminate the job because of an error. RA+2 through RA+77 hold SCOPE control card information directing current processing and information for the loader.

ACCOUNTING (JOB AND SYSTEM DAYFILES)

SCOPE maintains on mass storage the job dayfile, a chronological accounting of each job run, which is automatically printed at job termination. It contains a copy of all control cards processed, equipment assignments, diagnostic messages, central and peripheral processor time used, and the date and time of day associated with each processing event relative to the job. The job dayfile is maintained entirely by the system and is not accessible to user jobs, but jobs can send messages to the dayfile.

SCOPE also maintains a system dayfile, a record of every processing event in the system; it relates to all jobs in the system. It contains all information in the job dayfiles plus other relevant system messages. The system dayfile can be printed, punched on cards, or copied to tape at the operator's request. This system file is named DAYFILE and exists as a permanent file.

OPERATOR/SCOPE COMMUNICATION

The computer operator communicates with SCOPE through the keyboard of the display console, and SCOPE transmits messages to the console's twin screens. Several displays can be requested, and certain kinds of messages are associated with each. The displays most often requested are job status, input, output and execution queues, and dayfile displays. The job status display shows the progress of jobs currently at control points, the most current program message, and other information. The input queue display presents the status of jobs not yet assigned to the central processor for execution. The execution queue display presents the status of jobs which are in some stage of execution. The output queue display shows information about files from jobs which have completed executing and are waiting for output processing. Other displays include all equipment in the configuration, central memory content, and system tables content. The operator makes use of these displays to speed job processing. The analyst also uses them while debugging programs. In the dayfile display, the latest messages to the system dayfile are presented. Displays, operator requests, and other operator messages are described in the SCOPE 3.4 Operator's Guide.

The programmer communicates with the operator through parameters on control card and program statements such as the COMPASS macro MESSAGE and the FORTRAN PAUSE or COBOL STOP statements.

JOB PROCESSING AND DECK STRUCTURE 2

When the user prepares a job deck, he is structuring the job input file. The beginning of the file is the job card; the end is the end-of-file (EOF) card with multiple 6/7/8/9 punches in column 1. Between these cards, the end of each logical record is defined by an end-of-record (EOR) card with 7/8/9 multiple punches in column one, and optionally, a level number (section 3) in columns 2 and 3.

The control cards, each user program, and each set of data comprise individual logical records. Each record, except the last in a job, is terminated by an end-of-record (EOR) card. The last logical record is terminated by an EOF card, which acts as both an EOR and EOF indicator. Although only the EOF card is required at the end of a job deck, some programmers terminate their jobs with both an EOR and an EOF card as a matter of preference.

As a job is read into the system, it becomes known as a file with a logical file name equivalent to the identifier the user supplied on the job card as modified by SCOPE to ensure all job names are unique.

The first logical record in the job must contain all, and only, control cards. These cards direct components of SCOPE in performing user defined tasks. Basic functions of control cards are:

Identify the job and some of its characteristics

Request devices needed for execution

Call for compilation or assembly of a program

Direct the loading of jobs into central memory

Call for execution of a program

Specify exit paths, termination conditions, and procedures

Some cards, such as the FORTRAN compiler call, must specify the file to be compiled if the user does not wish the default file. Others, such as UPDATE and COPYN utilities, must specify both the file to be manipulated and the source of more details concerning the process or use existing default values. In the absence of specified files, the source of this information is assumed to be the job file itself. For example, when a COMPASS call is encountered in the control card record, the assembler assumes the next unprocessed logical record in the job deck is the file to be assembled, unless the COMPASS call contains an input file parameter.

Each control card is read and executed as it is encountered in the control card record. Consequently, logical records in a user job must be in the order they will be used when control cards are executed.

JOB FLOW THROUGH THE SYSTEM

The manner in which control cards establish user program handling is illustrated by following a sample job as it is processed. For example, consider a job to assemble and execute a program written in COMPASS, with the output to a line printer. The user gives the operator a tape to be used for output. In the sample job shown below, the tape has a label containing 1972 as the volume serial number.

The job would be structured as follows:

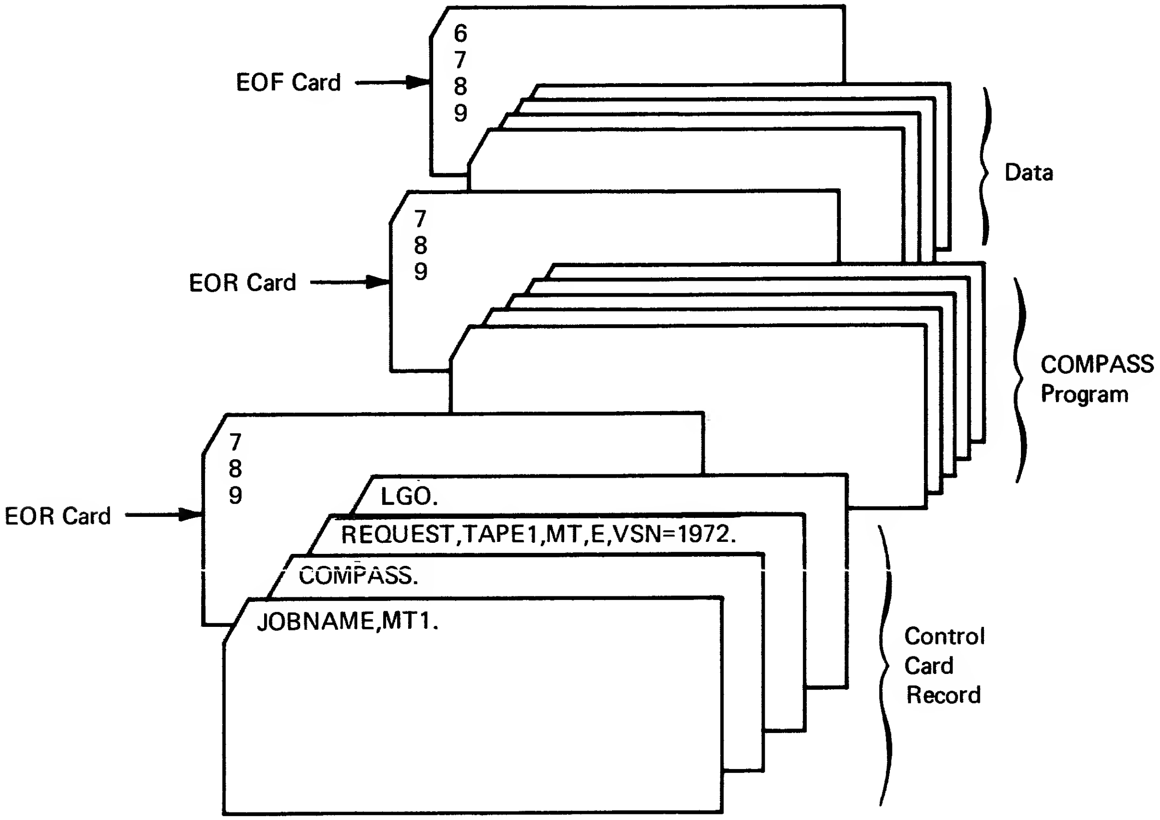
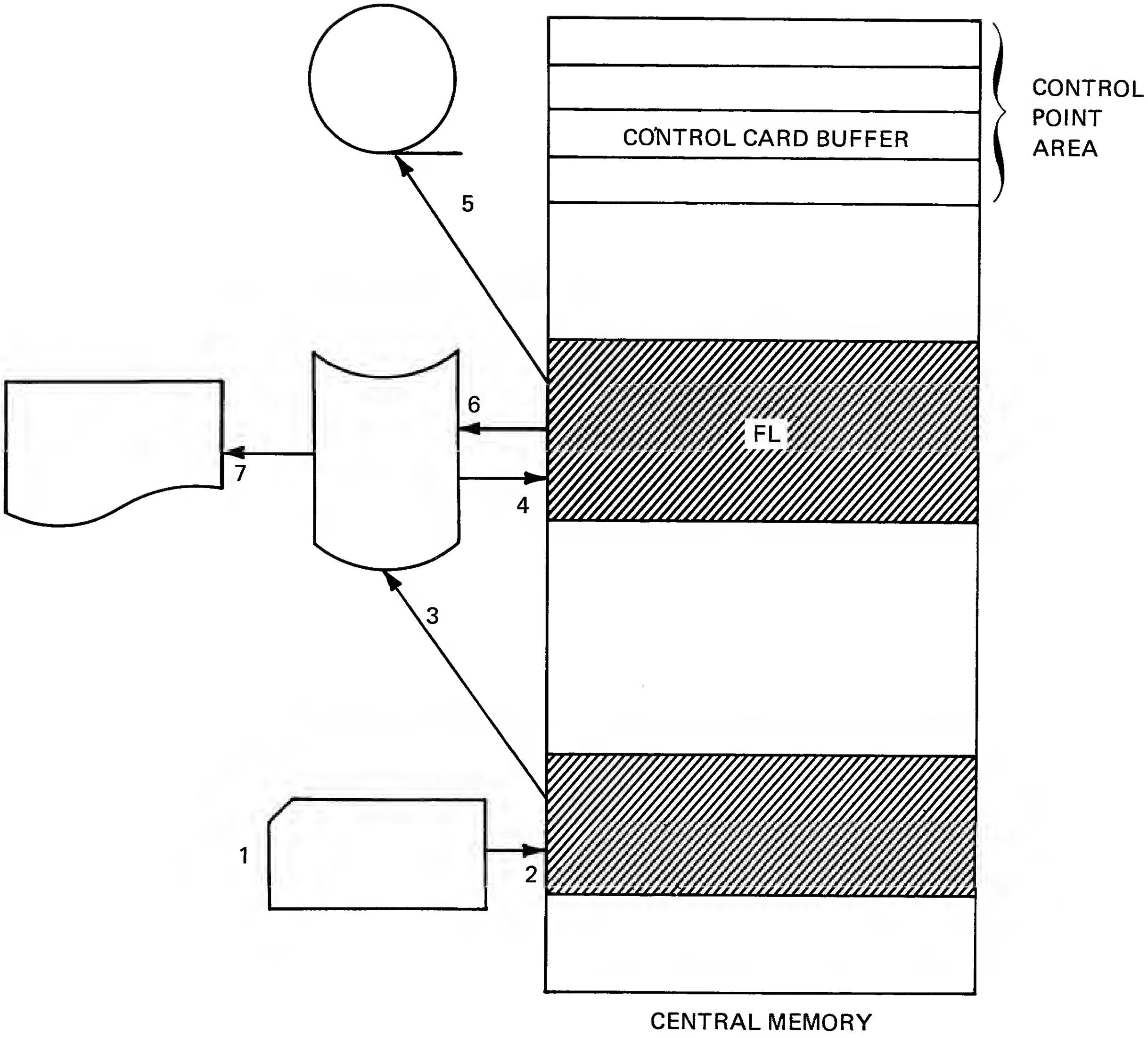


Figure 2-1. Sample COMPASS job

EXECUTION PROCEDURES

When the sample job is input through the card reader, SCOPE calls a PP routine to translate the job card, check the validity of its entries, and assign a priority to the job. Next the PP copies the job through a central memory input/output buffer onto mass storage. At this point, SCOPE identifies the job by its file name JOBNA01 (from the job card).



1 Job read into card reader
2 Job read through buffer onto disk
3 Job in mass storage input queue
4 Job assigned control point; goes into execution
5 Some output to a tape
6 Job assigned to output queue
7 Output to printer

Figure 2-2. Job Flow at Central Site

When the job is in the input queue of jobs awaiting execution, it comes under control of a SCOPE scheduling routine. The following factors are considered in assigning jobs to available control points: the priority entered with the job, available system resources such as central memory, direct access ECS, and tape units. A job descriptor table ordinal is assigned to the job; this ordinal is used to identify the job regardless of whether it is in central memory or not. Then the job file name is changed to INPUT.

SCOPE saves the original name of the job for later use. The file INPUT is positioned at the beginning of the second record (the user's program). The first control cards are read into a buffer within the related control point area in low core, and are ready for execution. The job is assigned a file named OUTPUT to collect job output.

Since the job card was processed when the job was read into the input queue, job control is advanced to the second card REQUEST. The VSN parameter provides the volume serial number for the tape label. SCOPE automatically assigns the tape if it is mounted. (If the installation does not choose the automatic assignment feature of SCOPE, the REQUEST card appears on the operator console; and the operator must assign the tape to the job manually.)

Next is the COMPASS control card, which directs assembly of the user's program. SCOPE requests the loader to load the COMPASS assembler into the field length. Control passes to COMPASS, to assemble the next logical record, the user's program, on the file INPUT onto a mass storage file called LGO. (For assembly or compilation, the user can designate files other than INPUT as an input file and other than LGO as binary output by entries on the COMPASS control card; but unless such alternative files are named on the assembly or compilation card—the COMPASS card in this case—INPUT and LGO are used by default.) COMPASS also writes a source language listing of the program onto a file named OUTPUT. Control proceeds to the next control card, LGO. At job termination OUTPUT is printed unless the user specifies otherwise.

The LGO card directs program execution. The loader loads the LGO file containing the user's program in object code into central memory and writes a map of this program onto the file OUTPUT; library subprograms required are loaded also. Control passes to the user's program for execution, input data is read from the third record on the INPUT file (user's data), and output is written on TAPE1 and OUTPUT.

As each control card is executed, it is copied onto the job and system dayfiles. The last card to be processed in the control card record on the INPUT file is the card with 7/8/9 multiple punches signaling the end of the job control cards. SCOPE writes the central and peripheral processor running times on the dayfile and copies this file to OUTPUT, which then is detached from the control point. The name OUTPUT is changed to JOBNA01 (the original job name) and TAPE1 is released so that the tape unit may be available for another job. INPUT, LGO and the dayfiles are cleared and released from SCOPE control. All equipment associated with the job is released from control point n and assigned to control point 0, where it can be requested by other jobs. The control point area and field length in central memory are made available for other jobs. When a printer is available, JOBNA01, containing the assembly language program listing, load map, output, and dayfile, is printed.

From the time a job is assigned to a control point and execution is completed, many other jobs are being executed. Each job is assigned a job descriptor table (JDT) ordinal when it is first assigned to a control point. If the scheduler routine swaps out the job (returns it to mass storage in its present state of execution), the JDT ordinal maintains the identity of the job when the control point association is lost. A job may be swapped out by the scheduler when a job with higher priority enters the system or when the job is delayed waiting for a resource such as a family pack. A job may be rolled out also, freeing central memory but retaining a control point, while awaiting operator action. The scheduler directs swapping and rolling, taking into consideration the relative needs of batch jobs and interactive jobs. When jobs are swapped or rolled into central memory, they resume execution at the point of interruption.

JOB TERMINATION

NORMAL TERMINATION

When a job is processed without error, normal termination activity begins upon reaching end-of-record (7/8/9) or an EXIT card in the control card record. First, execution time of the job is written onto the job dayfile, DFILEn, and on the system dayfile, DAYFILE. Then, DFILEn is rewound and copied onto the file OUTPUT. Next, OUTPUT and any other files on mass storage designated for output, such as PUNCH or PUNCHB, are rewound and placed in the output queue. OUTPUT is designated for the printer, and PUNCH (Hollerith) and PUNCHB (binary) for the card punch by disposition codes. These file names are then changed to the job name, and the assignment to control point 0.

Files on magnetic tape are rewound, (unloaded if the programmer requested save status), and released from the system. Permanent files are released from the job, returning to permanent file manager jurisdiction. For family pack files, all pertinent system information in central memory is copied to the pack label. All remaining files in central memory and mass storage associated with the job including INPUT, LGO, and DFILEn, are cleared and released. The job is released from the control point area.

All hardware devices assigned to a job are assigned to control point 0, so they can be reassigned to other jobs. At this point, only files in the output queue relating to the job remain. When an output device of the type requested by the file's disposition code is free, the file will be output through that device. When applicable, the job output is arranged in the following order:

Source language listing
Object listing
Load map
Executed program output (results)
Job dayfile

ABNORMAL TERMINATION

When an error occurs, SCOPE sets a flag indicating the error. If the error has been previously identified in the current job step by a call to RECOVR, control is returned to the user program for processing. Otherwise SCOPE continues with error processing.

A diagnostic message, reflecting the reason for abnormal job termination, is written to the OUTPUT file to accompany a standard dump of the exchange package, along with the contents of 100 (octal) words that both precede and follow the program stop location. SCOPE then clears the error flag, searches the control card record for an EXIT card, and executes the control cards that follow it. If no EXIT card is found, the job terminates as described under normal termination.

TERMINATION BY OPERATOR COMMAND

When the operator types in a DROP command, the job terminates prematurely. End-of-job procedures are initiated as described under abnormal termination.

When the operator types in a KILL command, the job terminates prematurely. All files associated with the job, including the OUTPUT file, are dropped regardless of name or disposition. Permanent files and family pack files are treated the same as for normal termination. The programmer will not receive a dayfile listing.

When the operator enters a RERUN command, the job is terminated and its INPUT file is returned to the input queue, so that it can be run later. The OUTPUT file is dropped, and a new output file is created. The job dayfile is copied to the new output file called a pre-output file, and becomes the OUTPUT file when the job is run again. The OUTPUT file for the rerun job will contain the dayfile from the previous partial run of the job and the output and dayfile from the complete run of the job.

Permanent files and family pack files for a rerun job are treated as for normal termination. All other files, regardless of name or disposition, are dropped.

In some cases, a job might perform a function which would make it impossible to restore conditions to their initial state before the job was run. For example, if a job writes on an existing permanent file, that information cannot be erased. When such a job is rerun, results are unpredictable. To avoid this condition, the system will set a no-rerun flag in the control point area to reject a RERUN type-in by the operator. The no-rerun flag will be set when the job has performed a catalog, purge, alter, rename, or extend of a permanent file; modified a permanent file or rewritten a family or sequential pack file.

Should a job be caught at a control point during a deadstart recovery, it is either dropped or rerun depending upon the no-rerun flag. If possible, the job is rerun; however, if the flag indicates no rerun, the job will be dropped and an appropriate message added to its dayfile. Any job swapped out during a deadstart recovery will be given a message indicating that recovery was performed.

JOB DAYFILE

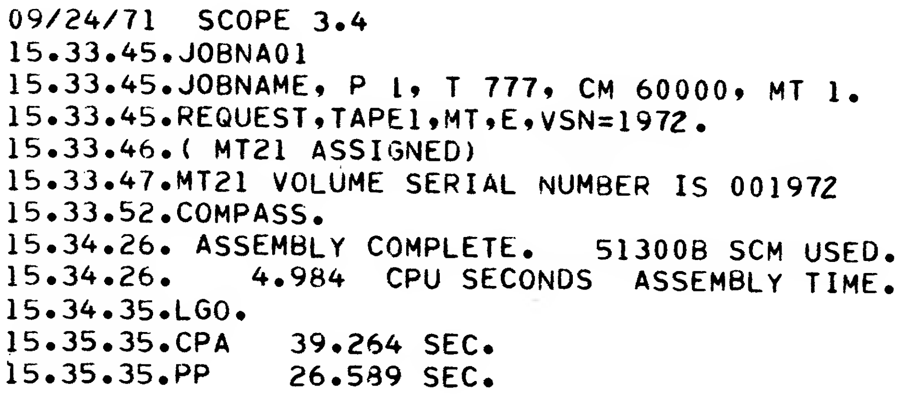
The job dayfile output with each job records the history of execution.

The first line shows the date the job was run along with the operating system and machine used for execution. Each control card is listed with the time it was encountered. Error messages produced by execution of a control card appear here. Operator or system actions affecting the job, such as a verification of equipment assigned or a job rollout, are recorded.

The end of the dayfile shows the number of seconds (decimal) of central processor and peripheral processor time used. The installation may also choose to show input/output time.

The programmer can cause information to be sent to the job dayfile by using the COMMENT card in the control card record or the MESSAGE macro in a COMPASS program. Several other language processors also allow messages to be sent to the operator or to the dayfile.

Figure 2-3 shows a dayfile from a job similar to that of figure 2-1.



```
09/24/71  SCOPE 3.4
15.33.45.JOBNA01
15.33.45.JOBNAME, P 1, T 777, CM 60000, MT 1.
15.33.45.REQUEST,TAPE1,MT,E,VSN=1972.
15.33.46.( MT21 ASSIGNED)
15.33.47.MT21 VOLUME SERIAL NUMBER IS 001972
15.33.52.COMPASS.
15.34.26. ASSEMBLY COMPLETE.   51300B SCM USED.
15.34.26.    4.984  CPU SECONDS  ASSEMBLY TIME.
15.34.35.LGO.
15.35.35.CPA    39.264 SEC.
15.35.35.PP     26.589 SEC.
```

Figure 2-3. Sample Dayfile

PROGRAM EXECUTION CARDS

A user can call a program for execution by using a control card that names the file containing the program. When such a card is processed, SCOPE searches a table containing the names of all user created files assigned to the job. When the specified file name is found, the file is rewound (rewinding is controlled by installation default and user override) and all programs on the file are loaded into central memory and executed.

If the file is not located, SCOPE searches the system libraries for an entry point of that name. If found, the program containing that entry point is loaded and executed. If no such entry point is found, the job is terminated.

Programs created by users and programs that are part of SCOPE are both referenced by the program execution card, in the following format:

```
lfn,list.
```

Only lfn is required; it is the logical file name of the file to be loaded, or of an entry point in a library program; it must be 1 to 7 letters and numbers, beginning with a letter and followed by a terminator if no parameter list follows.

The optional entry, list, is composed of parameters that depend on the program to be executed and are referenced by that program. A separator follows lfn and each succeeding item in the list. The last parameter must be followed by a terminator.

SEPARATE LOAD AND EXECUTE

The loader of SCOPE 3.4 differs from that of earlier SCOPE systems, as explained in the reference manual for the loader. These differences affect the sequence of control cards involving the loader, and consequently, the structure of records in the job decks. If operations are other than a call for load and execution of a program on the NUCLEUS system library (section 6), or load and execution of an object program created or attached by the current job, the LOADER Reference Manual should be consulted.

The LOAD control card specifies a file to be loaded but not executed; the EXECUTE card completes the load if necessary and initiates execution.

```
LOAD(lfn)

EXECUTE(lfn)
```

These two cards are the equivalent of:

```
lfn.
```

COMPILER OR ASSEMBLER CALLS

Names that should be used for lfn on the program execution card to assemble or compile a user program are listed below:

Source Language	**lfn**
FORTRAN Extended	FTN.
FORTRAN (RUN)	RUN.
COBOL	COBOL.
ALGOL	ALGOL.
COMPASS	COMPASS.
SIMSCRIPT	SIMS.
SIMULA	SIMULA.
BASIC	BASIC.
SYMPL	SYMPL.
SORT/MERGE	SORTMRG.
PERT/TIME	PERT66.
APT	APT.
QUERY/UPDATE	QU.
FORM	FORM.
QUIDDL	QUIDDL.

On cards requesting assembly or compilation, the list parameters are used for such functions as:

Naming the file to which the program is to be translated in object code (default name LGO)

Naming the file containing the program to be assembled or compiled (default name INPUT)

Producing source language or object code listings of the program (listing options such as S in FORTRAN)

Punching the file on binary cards (change default LGO to PUNCHB)

The following card requests compilation of a FORTRAN Extended program from a file called STANLEY onto a file named OLIVER.

```
FTN(I=STANLEY,B=OLIVER)
```

Load and execution of that program would result from:

```
OLIVER.
```

The list parameters associated with FORTRAN Extended as well as other source languages are described in detail in the respective reference manuals.

After compilation, loading and execution are normally requested.

THE LGO FILE

The object code output by compilers and assemblers is written, by default, to a file named LGO. The control card LGO used in many job decks is no different from any other call for program loading and execution discussed above. When SCOPE cannot match a control card with its list of control cards, it assumes the card is calling for program execution. It searches the user's files, then the system entry point lists, for a corresponding name.

Rewind before loading is controlled by installation default. At most installations, rewind occurs automatically before loading. In a straightforward compile and execute job, the user need not rewind LGO or its equivalent. If more than one program is written on LGO during a single job, however, user manipulation of LGO may be required.

For example, a combined program has the control cards:

```
FTN.
COMPASS.
```

Both compilers write to LGO. An execution call rewinds the file:

```
LGO.
```

The assembled output of both programs are loaded together.

To execute assembled output of independent programs. the following control cards can be used:

```
EG1.
FTN.
LGO.
REWIND(LGO)
COMPASS.
LGO.
```

Alternately, independent programs can be executed with control cards similar to:

```
EG2.
FTN(B=AAA)                 Assembled program on file AAA, not LGO
COMPASS(B=BBB)             Assembled program on file BBB, not LGO
AAA.                       Rewinds, loads, executes file AAA
BBB.                       Rewinds, loads, executes file BBB
```

Deck structure after control cards must correspond to the SCOPE logical record requirements defined by the control card order. In EG1, the deck must be:

```
Control cards
7/8/9
FORTRAN program to be compiled
7/8/9
FORTRAN data used during execution
7/8/9
COMPASS program to be assembled
7/8/9
COMPASS data
6/7/8/9
```

In example EG2, required deck structure is:

```
Control cards
7/8/9
FORTRAN program to be compiled
7/8/9
COMPASS program to be assembled
7/8/9
FORTRAN data
7/8/9
COMPASS data
6/7/8/9
```

EXAMPLES OF JOB DECK ARRANGEMENTS

The order in which SCOPE control cards are arranged within the control card record depends upon the purpose of the job and the programs it contains. The following examples illustrate typical arrangements. Automatic rewind before a load is assumed.

JOBA requests a tape file named SALLY, and loads and executes an object program from that file:

```
JOBA,MT1.
REQUEST,SALLY,MT,VSN=123456.
SALLY.
6/7/8/9
```

JOBB, containing a FORTRAN Extended program on Hollerith cards, compiles, loads and executes that program.

```
JOBB.
FTN.
LGO.
7/8/9
FORTRAN Extended Program
6/7/8/9
```

JOBC, containing a program on binary cards, loads and executes that program:

```
JOBC,CM43000,T500.
INPUT.
7/8/9
Program on Binary Cards
6/7/8/9
```

JOBD compiles and executes a FORTRAN Extended program and executes this program with one set of data, and then with another:

```
JOBD.
FTN.
LGO.
LGO.
7/8/9
FORTRAN Extended Program
7/8/9
First Data record
7/8/9
Second Data record
6/7/8/9
```

FILE STRUCTURE AND PROCESSING 3

In the SCOPE system, all information defined to the system is considered to be either a file or a part of a file. This section describes the concepts and terminology underlying the file organization and the structuring of data. SCOPE activities related to the creation, processing, and disposition of files also are included.

The SCOPE logical records discussed below are equivalent to Record Manager S type records.

LOGICAL FILE STRUCTURE

Each file is known by its logical file name (lfn). The name consists of one to seven letters and numbers, the first of which must be a letter. The internal central memory representation of a logical file name consists of its literal value in display code, left justified, and zero-filled in bits 59 to 18 of the central memory data word.

A file is defined to be information through an end-of-file (EOF) indicator. This indicator takes several forms, depending on the storage medium. The COMPASS macro WRITEF writes an EOF indicator appropriate to the file storage.

> On cards, an EOF is indicated by the presence of a 6/7/8/9 multiple punch in column one. Alternately, it may be a card with a 7/8/9 multiple punch in column one and a SCOPE logical record level number 17 in columns two and three.
>
> On magnetic tapes, files in standard SCOPE format have a 48-bit EOF marker containing a level 17 value. On magnetic tapes in other formats, EOF indicators are defined by tape marks.
>
> On mass storage, an EOF is represented by a sector with no data but a level 17 byte.

End-of-information (EOI) is the end of data. Some files may have more than one EOF but no file may have more than one EOI.

Files on cards, mass storage devices, and magnetic tapes in SCOPE standard format can be divided into SCOPE logical records. Level numbers 1-17 (octal) associated with each SCOPE logical record marker define the hierarchy structure of a file.

An end-of-record (EOR) marker on mass storage or SCOPE tapes has the same format as an EOF marker, but it contains a different record level number. On cards, an EOR is indicated by a 7/8/9 multiple punch in columns one, with level number, if any, in columns two and three. If a level number has only one digit, it may be punched in column 2. The highest record level number is 16.

The lowest level within a file is 0. It is associated with a single SCOPE logical record. A higher level defines a set of records consisting of the SCOPE logical record at that level plus all preceding records at a lower level. By this logic, the level 17 EOR defines an entire file containing other records. SCOPE logical records within a file are assumed to be level 0 unless specifically identified otherwise. The COMPASS macro WRITER allows an EOR of any level except 17 to be written.

Level numbers can be used to organize data in a file. Using the COMPASS macros SKIPF, READSKP and SKIPB, SCOPE logical records with particular level numbers can be accessed.

A level of 16 should not be used in a program that includes a request for a checkpoint dump. (See section 10). This level number is used uniquely by the Checkpoint/Restart program.

Level 15 is used by COBOL 3 to denote end-of-file; this usage has been discontinued in COBOL 4.0.

Files may be transferred from one device to another without destroying the logical file structure. (The SCOPE logical record concept is defined for all devices except magnetic tapes in S or L tape format).

On mass storage or SCOPE format tapes, the concept of a physical record unit (PRU) maintains the logical integrity of a file.

The physical record unit size is the largest amount of information that may be accessed during a single physical read or write operation for a given device. SCOPE logical records are written as one or more PRU's, the last of which is a short PRU or a zero-length PRU. If user data does not fill the last PRU in a SCOPE logical record, the EOR marker is appended to the data and the remaining PRU space is ignored. The foregoing defines a short PRU. On the other hand, a zero-length PRU must be created if the EOR marker cannot be accommodated in a PRU with user data. A zero-length PRU contains only system information such as the level number.

Level 17 EOF markers always are written as zero-length PRU's on magnetic tapes and mass storage. In COMPASS they can be produced by programmer calls to the macro WRITEF.

MAGNETIC TAPE FILES

A single reel of magnetic tape is known as a volume. A volume set can consist of:

—a single file reel of one file on one reel

—a multi-file of more than one file on a single reel

—a multi-reel file of one file extending over more than one reel

—a multi-reel multi-file of more than one file extending over more than one reel

All information on a magnetic tape begins after a physical reflective spot known as the load point. When this is sensed by a photoelectric cell, the tape is at its load point.

Subsequent information may be a label or the file data. Data may be in SCOPE, S, or L tape format on a 7-track tape divided into records and files. On 9-track tape, data may be in SCOPE or S format. Labels may exist on both 7 and 9 track tapes.

Beginning-of-information and end-of-information are dependent on the tape structure and format. Both are logical delimiters of data. The load point is the start of data for both unlabeled and labeled tapes. On a single reel labeled file, a REWIND reverses the tape to the load point, then skips forward to the file data following the label when a forward motion function is issued. End-of-information, which signifies no further file data exists, can be indicated only by the appearance of a trailer label field.

A tape mark is a short record used on SCOPE tapes to separate label groups and files from label information. On S and L tapes, it may also separate files in addition to separating label groups. Interpretation of multiple tape marks depends on the tape format. Contents of the tape mark record differ with tape density, as detailed in the label appendix. These tape mark records are written by SCOPE routines. On S and L tapes these records may be written by the COMPASS macro WRITEF.

Another physical reflective spot, known as the end-of-reel marker, appears near the end of all tapes. It warns the software to initiate end-of-tape procedures.

An unlabeled 7-track tape recorded at an installation-defined default density in SCOPE format is the default tape characteristic assumed by the system. Any other tape density, format, or label must be explicitly declared by a REQUEST or LABEL card. Unlabeled tapes always must be requested by a REQUEST card. A labeled tape can be requested by either a REQUEST card or a LABEL card. If both cards are used, parameter conflicts are resolved by the acceptance of values first encountered. A REQUEST card followed by a LABEL card is acceptable; LABEL followed by REQUEST for the same file will cause abnormal job termination.

DATA FORMAT

SCOPE 3.4 is capable of processing:

SCOPE tape–standard SCOPE format

S tape–stranger tape

L tape–long record stranger tape (7-track tape only)

Each can accommodate binary or coded data.

The reference manual for Record Manager contains detail regarding file structure and blocking. Generally, Record Manager equates the term block with a physical record on S and L tapes and with a SCOPE logical record (discussed below) on SCOPE tapes or on mass storage devices.

On 7-track tapes, SCOPE format is assumed unless S or L is explicitly declared on a REQUEST card or by the F parameter on a LABEL card. On 9-track tapes, SCOPE format is assumed unless S is explicitly declared on a REQUEST card or by the F parameter on a LABEL card.

Tape blocks are composed of information separated by inter-record gaps.

SCOPE TAPES

SCOPE tapes are the system standard. Data is written in tape blocks holding a physical record unit defined to be the contents of 512 central memory words of binary information or 128 central memory words of coded information. Coded information is written on tape in external BCD format for 7-track tape only. On 9-track tape, data is written in packed (binary) mode for both coded and binary operation.

SCOPE logical records on SCOPE tapes are terminated by markers generated in the PP when a tape is written and removed when the tape is read. Only the data passes from the tape to the user buffer in central memory.

For coded data being output on 7-track tape, the PP converts display code to internal BCD codes if a 6684 converter is not available. The tape controller converts internal BCD to the external BCD codes recorded on the tape. In the 63-character set, display code characters 33 and 63 both convert to an external BCD 12. In the 64-character set, display code characters 33 and 00 both convert to an external BCD 12. However, if the last two characters of a central memory word have a display code representation of 0000 (the end-of-line delimiter byte), they become an external BCD 1632.

For 7-track coded tapes being read in, the tape controller converts external BCD to internal BCD codes. The PP converts the internal BCD to display codes (if a 6684 converter is not available) before transferring data to the file buffer. On input, the external BCD 12 is converted to a display code 33 (zero). The end-of-line delimiter byte, which must occur at some multiple of 5 bytes, is converted to a 0000 display coded end-of-line byte.

Peculiarities for coded tape for the 63-character set:

		OUTPUT		INPUT	
	Display Code	Internal BCD	External BCD	Internal BCD	Display Code
	00	16	16	16	00
	33	00	12	00	33
	63	12	12	00	33
Line Terminator	0000	1672	1632	1672	0000

Display code 00 is not a valid character; display code 63 (colon) is lost. Line terminators (byte of all zeros in lowest byte of a central memory word) will not result in the loss of zeros.

Peculiarities for coded tape for the 64-character set:

	OUTPUT			INPUT	
	Display Code	Internal BCD	External BCD	Internal BCD	Display Code
	00	12	12	12	33
	33	00	12	12	33
	63	16	16	16	63
Line Terminator	0000	1672	1632	1672	0000

Display code 00 (colon) is lost; display code 63 (period) is now a valid character. An exception exists when up to 9 zero characters precede a line terminator. They are changed in the PP buffer to 63B. On tape, they result in external BCD 16. When tape is read, a 63 preceding a line terminator is converted to display code zero. This substitution ensures preservation of all zeros preceding a line terminator, regardless of the graphic character set used.

Appendix A contains the conversion tables for these codes. Conversion is performed by a 6684 if it is part of the hardware configuration.

In binary mode, the SCOPE logical record terminator marker contains the following information on 7- and 9-track tape; a 4-bit level number is right justified in the L field.

47	35	23	11	5 0
5523	3552	2754	00	L

The marker immediately follows record data if it can be contained within the tape block. Otherwise, it is written as the only information in the following tape block.

In coded mode, 7-track tape, each SCOPE logical record is terminated by eight characters in external BCD.

47	4 0
Blank (Reserved for Future System Use)	

Level Number, in Binary

Since coded operations for 9-track tapes are performed in packed (binary) mode, the logical record terminator is the same as the above diagram.

The level number is the low-order 5 bits of the last character. The upper 2 bits of this character are always zero except for level zero which is represented by 010000 (binary). For example, in external BCD, level five would be represented by 2020202020202005 and level zero would be represented by 2020202020202020.

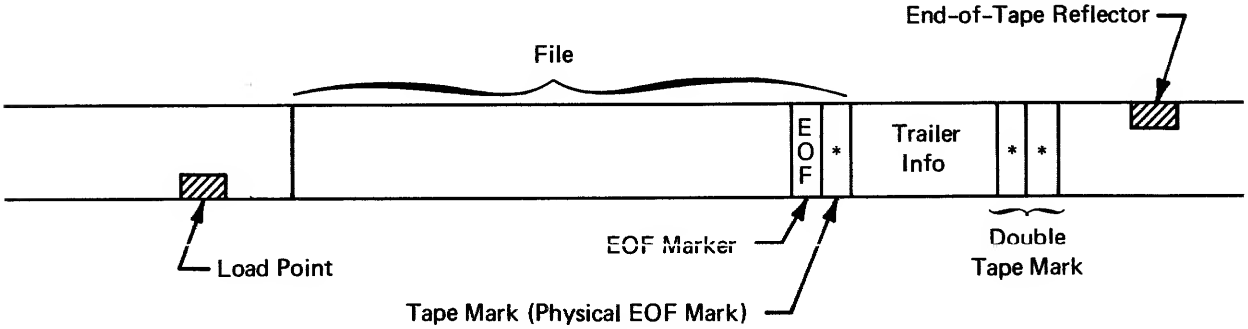
As with the binary marker, a record terminator marker is appended to the record data, if possible, or written as the only information in the following tape block.

In both binary and coded mode, a level number of 17 indicates an end-of-file. EOF marks are always in a separate tape block. An EOF is written in response to a user request to close a file or write an EOF mark.

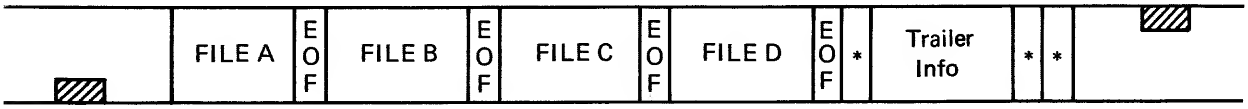
When a file is closed, rewound, or unloaded, SCOPE appends four items: a tape mark, trailer information identifying the file, and a double tape mark.

On a labeled or unlabeled SCOPE tape, a physical EOF mark followed by an EOF trailer label and a double tape mark signals that no further information exists. End of information is not defined for unlabeled S and L tapes.

Unlabeled SCOPE tape containing one file:



Unlabeled SCOPE tape containing four files:



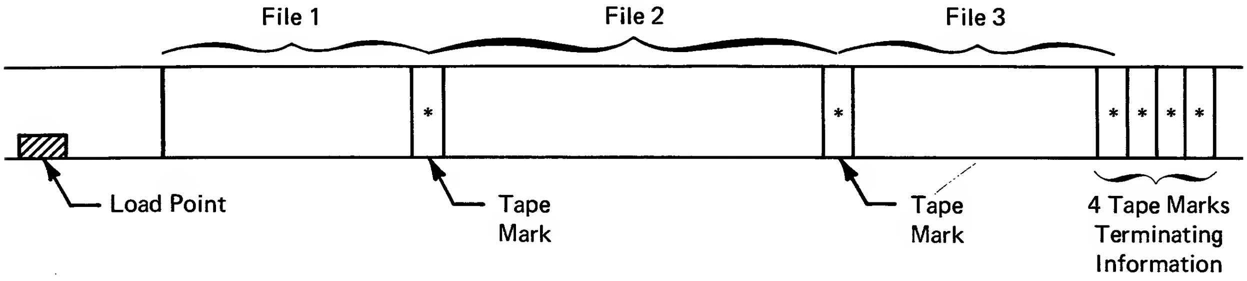
S AND L TAPES

S tapes contain physical records ranging in size from 8 to 5120 decimal characters. SCOPE considers a 6-character record on a 604 or 607 tape unit and a 7-character record on a 657 or 659 unit to be noise and does not process it. Specific record size must be stated by the user. In COMPASS, the MLRS (maximum logical record size) and UBC (unused bit count) fields in word 7 of the FET must be declared. MLRS declares the maximum number of 60-bit central memory words in the record. The last word may not be full of data since S tape records are measured in 6-bit characters, instead of words. UBC must declare the number of bits in the last transmitted word that are not used. This number must be a multiple of 12.

L tapes contain physical records ranging in size from 8 characters to an upper limit specified by the size of the buffer for the tape. Again, the user defines the record size in his program, with the MLRS and UBC fields.

Neither S nor L tapes contain SCOPE logical records of various levels as do SCOPE format tapes. The only records are the physical ones, and the file is that physically delimited by tape marks. The last file on an unlabeled S or L tape is terminated by four tape marks, but these are not recognized as EOI in the same sense as a label. The programmer must depend on the file marks or marks within his data. On a labeled S or L tape, an EOF1 replaces the second terminating tape mark.

Unlabeled multi-file S and L tape:



7-TRACK VS 9-TRACK TAPES

Both 7-track and 9-track ½-inch magnetic tape can be processed by SCOPE. Parameters on REQUEST and LABEL cards differ for recording densities, data format, and character conversion. Otherwise, label characteristics and tape usage are the same for both. A modular magnetic tape controller (MMTC) can be used to control 657 and 659 tape units.

In 9-track conversion (coded) mode, for S tapes, the MMTC matches a 6-bit character with a corresponding 8-bit equivalent. This conversion is performed when tapes are read or written. In packed (binary) mode, for S tapes, the MMTC writes each pair of 12-bit bytes (24 bits) as three 8-bit characters; conversely, for a packed mode read, the MMTC packs three 8-bit characters which are transmitted from the MMTC as two 12-bit bytes.

Seven track tapes are processed by 604, 607 and 657 tape drives. Data can be recorded in three densities:

LO	200 bpi	(low)
HI	556 bpi	(high)
HY	800 bpi	(hyper)

Installation-defined default densities will be used for reading unlabeled tapes and writing both labeled and unlabeled tapes in the absence of explicit declaration. The density of the label determines data density for reading labeled input tapes. However, it is always advisable to specify density because of the reading peculiarities of the tape drives. A tape label can be read at an incorrect density without causing a parity error; longer data blocks read at an incorrect density will cause parity errors.

On a REQUEST card, MT explicitly defines this tape as 7 track; LO, HI, or HY provides an implicit definition. On a LABEL card, 7-track is assumed unless 9-track is specifically declared.

Nine-track tape is processed on a 659 tape unit. Recording densities are:

HD	800 bpi	(high density)
PE	1600 bpi	(phase encoded)

On a REQUEST card, the NT parameter specifies a 9-track tape. On both REQUEST and LABEL cards, a density specification of HD or PE implicitly specifies a 9-track tape. Under hardware control, these tapes are always read at the density at which they are written regardless of declared density. Writing is done at an installation default density unless HD or PE is declared.

Conversion of 9-track tapes between 6-bit display code values used by CONTROL DATA central memory and a 64-character subset of 8-bit ASCII or EBCDIC tape code proceeds according to an installation defined parameter unless the user specifies otherwise (see conversion tables in appendix A).

US	ASCII conversion
EB	EBCDIC conversion

When SCOPE format 9-track tapes are written or read, data is not converted or manipulated in any way by the system. When SCOPE format tapes are read or written each central memory word to be processed is assumed to contain 60 bits of valid data. On SCOPE format tapes, partial central memory words cannot be read or written.

For 9-track S tapes, packed mode input/output is performed without any type of data conversion or manipulation on the part of the system. In conversion mode, however, data in the users buffer is assumed to consist of a string of 6-bit display code characters which will be mapped into 8-bit characters when written to tape. The 8-bit characters may be a subset of either ASCII or EBCDIC, as specified by the user on either the REQUEST or LABEL card or macro. Conversion from 8-bit to 6-bit display code is also performed when reading a 9-track S tape in conversion mode.

Multi-file tapes can be either 7- or 9-track tapes. Checkpoints can be taken on either. L tapes are not supported on 9-track devices.

TAPE LABELS

Labels on a tape consist of 80-character records that identify the reel of tape and the files it contains. They are the first records after the beginning-of-reel mark. Labels can appear before data on both 7-track or 9-track tapes.

SCOPE considers a tape to be labeled if the first record is in one of the following formats: any other tape is considered to be unlabeled.

> Standard 6000 series label format with a VOL1 identifier as the first four characters. ANSI compatible labels are identified on REQUEST cards as U labels, and on LABEL cards by the absence of Z or Y type labels. Z labels are processed in the same way as ANSI labels with the following exceptions: Z labels are not written by SCOPE 3.4; the Z parameter is treated as U for output tapes.
>
> When labels are read, the data density of the file is set according to the value of character 12 of the VOL1 label.
>
> CDC 3000 Series computer label format. They are identified as Y labels on REQUEST and LABEL cards.

Labeled tapes provide the following advantages for the user:

When a write ring is left inadvertently in an input tape reel, software checking will ensure that an unexpired label is not over-written without the express permission of the operator.

The number of blocks written on a file will be recorded in the file trailer label, as well as in the job dayfile. On subsequent file reading, the count serves as additional verification that data was read properly.

The reel number field of the label ensures processing of all reels in the proper sequence.

Multi-file reels with ANSI labels can be positioned by label name, rather than by file count only.

The volume serial number of any ANSI label read or written will be recorded in the dayfile.

Overall job processing time is reduced when the system can use the VSN field to locate and assign a tape to the requesting job without operator action at the keyboard.

The maximum benefit from the SCOPE tape scheduling and automatic tape assignment features can be derived only if all magnetic tapes used at an installation are labeled.

The standard 6000 series label format conforms to ANSI standard X3.27-1969. ANSI defines 10 types of labels by the first four characters in the label (VOL, UVL, HDR, etc). Appendix E details these labels. SCOPE requires the use of only four label types: VOL1, HDR1, EOF1, and if a file continues to another volume, EOV1. Other ANSI label types will be accepted by SCOPE for reading or writing when extended label processing capabilities are requested through the XL bit of the file environment table, as explained in section 11. However, all manipulating of such labels must be done by user code.

Under the current ANSI standard, density of label data is the same as that of subsequent data. Earlier standards, and therefore labels written by operating systems prior to SCOPE 3.4, allowed data recording density to be specified by character 12 of the VOL1 label. Tapes with earlier standard labels must be identified as Z labels on REQUEST or LABEL card if the label and data densities on the tape are not the same.

3000 series labels are the same as standard labels generated by the 3000 series computers. A file header, and end-of-file and end-of-tape trailer labels are defined.

Both ANSI and 3000 labels are written at the same density as subsequent file data. Default density is installation defined.

Information in label fields originates in several ways:

Defaults written by the SCOPE system when Y or U appears on REQUEST card

LABEL card parameters

COMPASS language manipulation of label buffers with extended label processing feature

Compiler language statements, such as COBOL LABEL statements

SCOPE generates labels with default values if the user does not provide otherwise.

ANSI STANDARD LABEL FIELDS

The four ANSI labels required are used as follows. Tape marks separating items are completely system controlled.

VOL1 — Must be the first label on a labeled tape. Each volume in a volume set must be identified as VOL1; a field in the following HDR1 label gives an actual reel number.

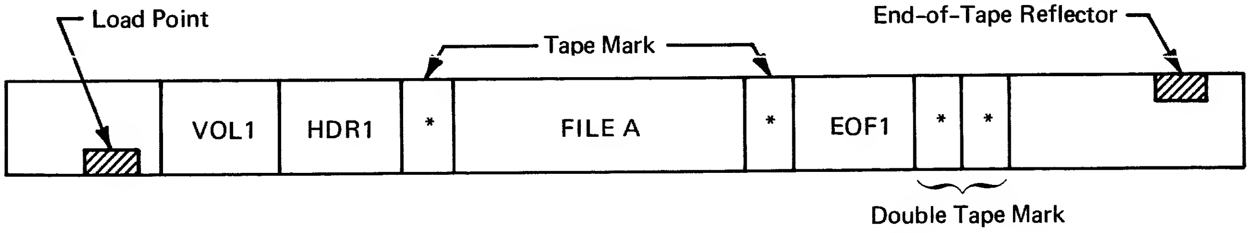
HDR1 — Required label before each file or continuation of a file on another volume. It is preceded by a VOL1 label or tape mark. Each file must have a HDR1 label which specifies an actual position number for multi-file sets.

EOF1 — Terminating label for file defined by HDR1 label; the EOF1 label is the SCOPE end-of-information for the file. A single tape mark precedes EOF1. A double tape mark written after the EOF1 label marks the end of a multi-file set.

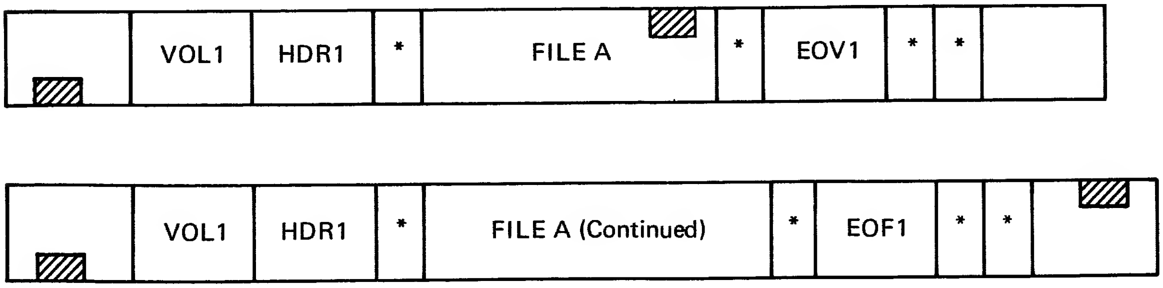
EOV1 — Required only if physical end-of-tape reflector is encountered before an EOF1 is written or if a multi-file set is continued on another volume. It is preceded by a single tape mark and followed by a double tape mark.

The structure of SCOPE tapes that results from these required labels is shown below. The label identifier and number is used to denote the entire 80-character label in these figures.

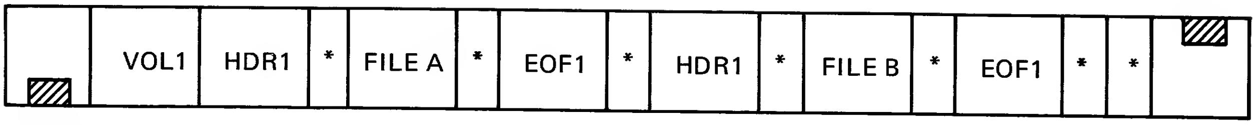
Single reel file:



Multi-reel file:



Multi-file reel:



Multi-file multi-reels in a volume set are also possible.

Tape label configuration that occurs when an end-of-file coincides with end-of-reel is given in Appendix E along with placement of optional labels in relation to required labels.

All required labels are checked by SCOPE on input and generated on output, unless the user issues contrary instructions. The NS parameter of the REQUEST card inhibits label processing. A REQUEST with a U parameter for a standard label tape results in the following default information in the required labels.

Volume header label:

Field	**Default**
label identifier and number	always VOL1
volume serial number	as typed at console or read from existing label
accessibility classification	blank giving unlimited access
owner ID	blank
label standard level	blank or 1 indicating ANSI standard

File header label:

Field	**Default**
label identifier and number	HDR1
file label name	blank
multi-file volume set identifier	VSN identifier
reel number for multi-volume file	0001 specifying the first volume in a set
position number for multi-file volume set	0001 specifying this is the first file in the set
edition number	0

Field	Default
creation date	current date
expiration date	current date
accessibility classification	blank
block count	000000
system code identifying operating system	blank

File trailer label:
Items and defaults same as HDR1 label except:

label identifier and number	EOF1
block count	number of blocks since last HDR label

Volume trailer label:
Items and defaults same as HDR1 label except:

label identifier and number	EOV1

With the exception of the label type identifier and number, any field can be changed through the LABEL card. A VSN card can supply the volume serial number if it is not given in the LABEL and REQUEST card, but such identification is not written on the tape.

TAPE IDENTIFICATION FOR AUTOMATIC ASSIGNMENT

This feature of SCOPE enables the operating system to assign a tape to the requesting job as soon as the operator mounts it on a tape drive. Automatic tape assignment is implemented by volume serial number (VSN) identification. Consequently, all magnetic tapes used at an installation selecting the automatic assignment option must have a VSN. This VSN may be supplied by the programmer or operator, and it may consist of a permanent field in a tape label or a temporary field in a tape processing table internal to SCOPE.

The VSN defines the volume of tape on which a particular file resides. The programmer associates this with a logical file name through:

REQUEST control card or function parameter

LABEL control card parameter

VSN control card

Only the VSN card allows multi-reel file identifiers or alternate tape reels to be specified. Use of the VSN card is recommended when a job's tape file requirements change frequently.

If more than one VSN parameter is given for a single file, SCOPE accepts the first encountered. Therefore, deliberate duplication provides the programmer with the ability to override, for example, a REQUEST function specification within a program without changing the program.

RANDOM ACCESS DEVICE FILES

Random access devices are drums, disks, and disk packs; generally, these random access devices are shared by many files and are known as public devices. A disk pack however, may be assigned to a single file attached to a single job, in which case it is processed sequentially similarly to a magnetic tape and not considered a random access device. A family disk pack, although a private device when attached to a single job, is a random access device and files are written the same as on public random access devices.

A file on a random access device may be of arbitrary length, and it may be segmented over more than one device. The data will be recorded in a logical sequence of record blocks which may be arbitrarily scattered about the disk or drum surface. SCOPE maintains a central memory table for each file, called the record block table (RBT), in which the sequence of allocated record blocks is defined. The end-of-information position is also defined in the RBT.

No physical distinction exists between binary and coded files recorded on these devices. Data from an integral number of central memory words in a user buffer is written to the device, and on reading, is returned to the buffer without transformation. A file may be written in binary and read as coded, or conversely. This practice is not recommended, as it will not work on other devices.

SEQUENTIAL AND RANDOM FILES

Files on random access devices may be written in random or sequential format. Sequential files are written or read from beginning to end. A program could skip records forward or back while reading; but the physical relation of the data, as recorded in the RBT, determines skip parameters. Although a sequential file may be written in arbitrary locations on a random access device, its integrity as a sequential file is not affected.

Most random files have associated indexes; direct access files are an exception. The index contains a relative PRU position for each SCOPE logical record in the file. The file beginning is equivalent to the start of the SCOPE logical record associated with the first index entry; the file end is equivalent to the end of the SCOPE logical record associated with the last index entry. Any record can be read by identifying it in the index without the need to skip records from some beginning file position.

If a random file is to be saved, the file index must be written as the last logical record on the file. A user may write the index or he may call the COMPASS macro CLOSE or CLOSE/UNLOAD to write the index. CLOSE automatically writes out an index for a random file if the file contents were changed by a write with the FET random bit set. A permanent file must also have EXTEND permission before the index can be made a part of the file.

USER DEFINED RANDOM FILES

SCOPE offers flexible handling of random access files. Single-level SCOPE name/number indexed files may be created and maintained using system macros READIN, WRITOUT, OPEN, and CLOSE; data record management at any level lower than a SCOPE logical record falls to the user.

READIN/WRITOUT may be used to create and maintain index contents during program execution without using OPEN/CLOSE to manage the index records. The user must manage his index records. They could be kept on a separate file, for example.

Multi-level SCOPE name/number indexed files may be created and maintained using READIN/WRITOUT and system macros OPEN and CLOSE plus a user generated sub-index management routine. A master index record would contain addresses of sub-index records interspersed throughout the file. The master index record would be processed by OPEN/CLOSE as is a single-level index record. The user routine would need to ensure that READIN/WRITOUT would reference the correct index or sub-index block.

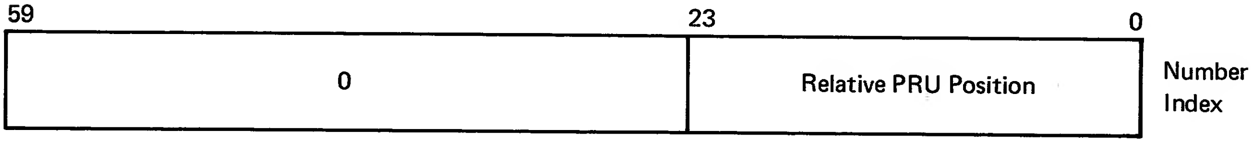
Other index formats may be defined by supplying a user routine to format and retrieve record names and mass storage addresses. Mass storage addresses may be computed on files containing fixed length records, provided the file is not ECS resident, since the addresses are in the form of a relative PRU count and the PRU size is fixed.

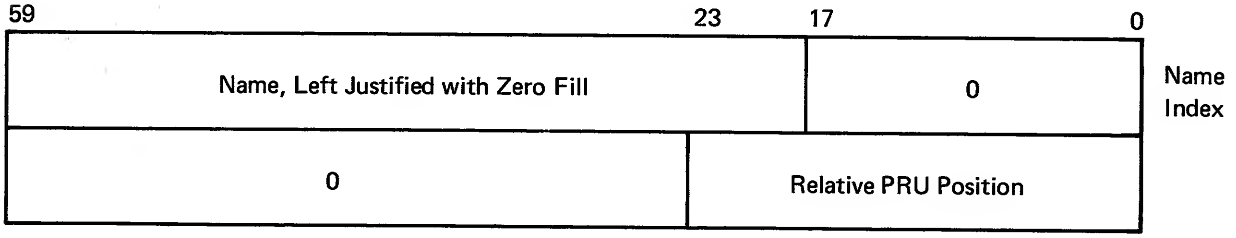
SCOPE NAME/NUMBER INDEX FILES

SCOPE indexed files may be created and accessed through macros READIN and WRITOUT which call the central processor routine IORANDM. They may be processed through COBOL, FORTRAN, and COMPASS language programs, and they may be declared as permanent files to be maintained on a public device.

SCOPE indexed files may be created, read, written, and rewritten using the COMPASS macros OPEN, CLOSE, READIN, WRITOUT, WRITIN, and WRITER. Management of a single index level is provided through macros OPEN and CLOSE.

The first word in the SCOPE index determines how the records are referenced. The index is generated through the WRITOUT macro (section 12). A positive non-zero value indicates reference must be by number; a negative value indicates reference can be by name or number. Number index entries are one word; name index entries are two words. The number of a record is equal to the relative position of the index entry for that record; the first entry in the index points to record 1, the second to record 2, etc. If a name index is used, the record name maybe 1 to 7 letters and numbers. The value of index word 1 is determined when the first record is written. The formats of index entries are shown below.





The smallest unit of information that can be indexed is a SCOPE logical record, and each SCOPE logical record must begin in a new PRU. For the most economical SCOPE index, SCOPE logical record length should be equal to an integral number of PRU's minus one word.

DISK PACK FILES

Disk pack drives can be used as follows:

System public devices to which files from all jobs can be assigned

Family packs assigned to a single job

Sequential packs containing a single file assigned to a single job

If a device is public, SCOPE will use it for many files assigned to many different jobs as the need arises. Permanent files can reside on these devices if the installation so designates.

If a device is private, it can hold a non-allocatable family or sequential pack. These packs can be used only by a single job. The main distinctions between these two types of packs are shown below:

	Family Packs	**Sequential Packs**
File format	Random or sequential	Sequential only
File capacity	63 files	1 file
Family size	1-5 packs (limited to number of drives available)	No limit on packs in file
Removability	All family packs must be mounted until job complete	First pack can be removed as processing continues on second pack drive if two drives have been requested, or on first drive with another pack mounted.
Method of request	RPACK card for manual assignment; REQUEST card with PK parameter	REQUEST card, DP parameter (This type of file may not be ECS buffered.)
Identification	Volume serial number (VSN)	Visual identifier (VID)

SEQUENTIAL PACK PROCESSING

A sequential pack contains a single sequential format file for use by a single job. No file size limit is imposed; the file can extend over many packs in the same way that a magnetic tape file can extend over many volumes. Sequential pack files, unlike family packs, are not limited to the number of units available, since only the portion of the file currently in process need be mounted. These packs can be dismounted during job execution when they are not in use.

Sequential packs are requested for use by the REQUEST card. The DP parameter indicates a sequential pack. A new pack is requested by:

```
REQUEST(lfn,DP,VID=nnnnnn,dt,N)    or    REQUEST(lfn,2DP,VID=nnnnnn,dt,N)
```

lfn — Logical file name

DP — Sequential pack identifier. If 2 precedes DP, the operator will assume a multi-pack file is involved and assign 2 units. The two assigned units must be the same device type.

nnnnnn — Visual identifier of 1-6 characters. The VID must refer to the first pack of the sequential set. Adherence to COBOL/ANSI standards for access of subsequential packs is handled by the compiler.

dt — Device type mnemonic:

AP	854 disk pack
AM	841 multiple disk drive

N — New pack indicator. Default is N.

Parameters after lfn are order independent; only lfn and DP are required. If VID is not supplied, the operator will supply it when the pack is mounted. The VID will be incorporated in the label of the pack to verify that packs assigned are those requested. Each pack will have a unique VID; the VID of the first pack of a multi-pack file will be used to identify all packs in that multi-pack file as well.

An existing pack is requested with:

```
REQUEST(lfn,DP,E,VID=nnnnnn,dt)
```

All parameters except dt are required for existing packs; they may be in any order.

When a file spans more than one pack, two pack drives can be requested with 2DP. Processing can continue on the second pack while the next pack is mounted on the first unit. The first unit is turned off when its pack is processed and must be turned on when the next pack is mounted. If multiple packs are involved, the operator is responsible for the order in which they are accessed. They must be accessed sequentially starting with the first in the series. If they are mounted out of order, the system will display a message and give the operator the option to continue processing or to mount another disk pack and recheck the label.

Operator action always is required to assign sequential pack units to the job. An * should not be used with the DP parameter.

Multipack files can be processed on one or two units by programmer request. During file processing, the operator is instructed to mount subsequent packs as necessary.

Examples:

The following cards request new packs:

`REQUEST(SEQFILE,DP)`	Default N; operator will supply VID
`REQUEST(SFILE,VID=12345,N,DP)`	Operator must mount pack having VID. 12345
`REQUEST(SEQUENT,2DP,N,AP)`	Operator must assign two 854 packs

Packs with existing files are requested in the following examples:

`REQUEST(SEQ,E,VID=222222,DP)`	System will verify operator assignment of pack with VID 222222
`REQUEST(SEQTL,2DP,E,VID=122221)`	First pack that operator assigns must have VID of 122221; system will request additional packs in file by VID.

MAGNETIC TAPE PROCESSING

Magnetic tape files to be used or created by a job must be explicity requested. Three control cards may be involved: REQUEST, LABEL, and VSN.

The REQUEST card can be used for all tape files – labeled, unlabeled, single file, or multi-file set. Parameters, in addition to specifying format and density, can specify processing for the file. Identifying the tape as input or output and the type of label is sufficient to initiate label processing and checking when the file is opened. The installation default options for unloading, label processing, and parity error processing may be overridden. A volume serial number parameter for the reel (or first reel in multi-reel file) will allow the system to assign the file automatically.

The LABEL card can be used in place of a REQUEST card for a labeled, single file reel and to write or check file header labels on single or multi-file reels. Parameters on the card establish label type and whether labels are to be read or written. Fields in file header (HDR1) labels will be written or checked according to the values on the card. If a multi-file reel is to be labeled, a REQUEST card must first establish the multi-file name, then a LABEL card may exist with the name and label field values for each file in the set. With the LABEL card, either a volume serial number or a label name may be given for identification for automatic tape assignment purposes. Automatic assignment by label name applies only when the read (R) parameter is specified on the LABEL card. The LABEL card also can be used to position to a particular member of a multi-file set.

EXTENDED CORE STORAGE USAGE

I/O BUFFERING THROUGH ECS

All sequentially accessed mass storage files may be buffered through ECS to avoid the costly access time of rotating mass storage devices each time a small amount of information is transferred.

In order to optimize the access to such devices, a larger amount of information is transferred between the device and ECS at the time of each access. Then regular (smaller) user transfers take place for each CIO call between ECS and the user's buffer in CM at a high rate of transfer and without involving any device positioning.

The information read ahead (input file) or waiting to be written (output file) is stored temporarily in an ECS buffer. The underflow and overflow functions for these ECS buffers are performed automatically by the system.

The I/O buffering scheme is suitable only for files that can be accessed sequentially, including permanent files.

The ECS buffers are requested on a file-by-file basis through the REQUEST card, or macro, or by an ATTACH card or macro and a different buffer size can be specified for each file if the standard buffer size is not desired.

Moreover an installation option allows the automatic allocation of an ECS buffer (standard size) to all files allocated by default and processed sequentially.

In any case, the data contained in an ECS buffer is written to a mass storage device only if the file is closed or exceeds the limit of the ECS buffer.

For optimum performance, the ECS buffer should be many times the size of the user's CM circular buffer. This will ensure that the system overhead associated with ECS buffer management is small compared to the time saved as a result of performing fewer device accesses. Suggested relative buffer sizes are:

CM Circular Buffer	ECS Buffer
201 - 401 octal words	4000 - 10000 octal words
501 - 1001 octal words	10000 - 20000 octal words
1001 octal words or greater	20000 octal words or more

For I/O bound programs using large CM circular buffers there is little advantage in using I/O buffering. In general, an I/O buffer can be used to reduce the CM buffer size while maintaining the high transfer rates associated with having large CM circular buffers. Throughput on I/O buffered files is primarily a function of the ECS buffer size, rather than the CM circular buffer size. Thus, rarely is it necessary to have a circular buffer greater than 1001 words when ECS buffering is employed.

If an unrecovered ECS parity error is encountered with the EP bit set, control is returned to the user program with the error noted in the code and status field of the FET. If the error occurs with the EP bit off, a GO or DROP decision is required of the operator.

ECS RESIDENT FILES

This facility is provided as an installation option selected when the system tape is built. Except for some specific applications where a faster, limited rotating mass storage device is needed, it is generally preferable to use the I/O buffering scheme instead of ECS resident files. I/O buffering allows an overall optimization of the system.

Nevertheless, under this option any non-permanent sequential or random file can be ECS resident with either the limit of the available ECS or the limit specified on the REQUEST card (whichever is reached first). If no EC parameter is present on the REQUEST card, the file will not exceed the default I/O buffer size specified at deadstart time. ECS resident files are requested on a file-by-file basis, the REQUEST card having the same format as the one used for buffer allocation with the addition of the device type mnemonic of AX.

ECS resident files and ECS buffered files may coexist.

When an overflow occurs, i.e. all ECS pages are allocated or the maximum file size is exceeded, an error code 10 (device capacity exceeded) is stored in bits 9-13 of the code/status field and control is transferred to the user if the EP bit is on, else the job is aborted.

Note: If ECS is turned OFF, all requests for ECS buffers will be ignored and the files requested on ECS will be allocated on other mass storage devices.

CONTROL CARDS 4

CONTROL CARD FORMAT

Except for the job card, all cards in the control card record of a job deck have the same general format. They begin with a keyword of 1-7 letters and numbers. Leading blanks may appear before the keyword. If only the keyword appears, it must be terminated with a period or a right parenthesis.

Parameters may follow the keyword. Order-dependent parameters must be in the order specified; other parameters may appear in any order. Parameter formats depend on the control card specified, as detailed in other parts of this manual. A terminator must follow the last parameter.

A parameter may be formed from a literal. A literal is a character string delimited by dollar signs. Any parameter field that includes characters other than letters, numbers or asterisk must be written as a literal. Blanks within the delimiters are retained. If the literal is to contain $, two consecutive dollar signs must be written. The literal $A B$$41$ is interpreted as A B$41.

Separators between parameters may be any of the following characters:

(, / =

Any character with a display code value greater than 44 except *) $. and blank

A blank is not considered a separator except after the keyword. Blanks in the control statement are ignored except in a literal or after a keyword. (Slash (/) and equals (=) also may be used to separate fields within a parameter, depending on the control statement.)

A terminator must complete each control statement. The period and right parenthesis are the only legal terminators. Any characters appearing after the terminator are considered to be a comment. Comments are not interpreted, but they are copied to the job dayfile.

As an example, consider a control card requesting the operator to assign reel number 4326 to a file named TAPE2. The keyword is REQUEST; parameters are TAPE2, MT (for 7-track magnetic tape), and VSN=4326. PUT WRITE RING IN is a comment to the computer operator. This card could be written in any of the following formats:

```
REQUEST,TAPE2,MT,VSN=4326. PUT WRITE RING IN
REQUEST(TAPE2,MT,VSN=4326) PUT WRITE RING IN
REQUEST TAPE2,MT,VSN=4326. PUT WRITE RING IN
```

Control cards that may be continued on another card are: LABEL and the permanent file functions. If a terminator does not appear on a card, column 1 of the following card is assumed to continue the previous card.

In the statement formats that follow, upper case letters indicate constants; lower case letters indicate values to be supplied by the user. In examples that follow, if a control card is referenced before it is fully explained in the text, a brief notation of its purpose appears beside the card.

PROCESSING WITH CONTROL CARDS

The control cards discussed in this section pertain to general job processing and control, including file manipulation. Control cards pertinent only to checkpoint/restart, permanent files, user libraries, and utility routines are discussed in the respective sections of this manual. UPDATE is detailed in the UPDATE Reference Manual, and loader control cards are discussed in the LOADER Reference Manual.

JOB IDENTIFICATION AND CONTROL

IDENTIFYING JOBS (Job Card)

A job is identified, certain resources are requested, and processing priority levels are established with the job card. This card must be the first in the job deck; any other card appearing in this position is presumed to be the job card and will be interpreted accordingly.

Job card format:

```
    xxxxx,Tt,CMfl,ECfl,Pp,Dym,MTk,NTk,TPk,CPp.
```

One parameter, the job name, is required on all job cards. Other parameters may be included to specify resources, priority levels, or processing time limitations. If these parameters are omitted, SCOPE automatically assigns the system default values established when SCOPE is installed. Parameters may be listed in any order following the job name.

SCOPE ignores all blanks and any unknown parameters that appear on the job card. However, when improper characters are used as variables with valid parameters, the job will be terminated. For example, parameters such as CMABC and DATA would cause job termination since CM must be followed by digits and D followed by two letters and digits.

SCOPE interprets all numbers on job cards as octal values unless this procedure is redefined by the system analyst when SCOPE is installed at the user's installation.

The required job name parameter is:

xxxxx Name the user assigns to the job to identify it to SCOPE. Any combination of characters, numbers, or letters can be used; the first character must be a letter. A name longer than five characters is truncated to five.

SCOPE automatically modifies the name of every job by assigning letters and numbers that differ for each job as the sixth and seventh characters. This ensures unique identification if a job is entered with a name duplicating that of another job already in process. For example, if two jobs are named JOBNAME, one may be processed as JOBNA23 and the other as JOBNA34. If a job name contains fewer than five characters, SCOPE fills with zeros all unused characters through the fifth, and adds unique sixth and seventh characters.

The optional parameters follow:

Tt

t is an octal value for the time, in seconds, which the user estimates his job will require the central processor. It must include the time required for assembly or compilation; it does not include time during which the job is in the input queue or in central memory but not using the central processor. If the job access to the central processor exceeds the value specified by t, the job will be terminated prematurely. (Use of the RECOVR feature allows results of execution to that point to be recovered before termination).

t may not exceed five digits. An infinite time can be specified by 77777; the job will proceed until completed even if it exceeds the installation maximum value for t. A time limit of 77777 should not be used indiscriminately as certain kinds of program errors, such as an infinite loop, can result in great waste in such cases.

The job card could be written in any of these formats:

```
jobname,T400.

jobname(T400)

jobname(T400.
```

CMfl

fl is the maximum field length (octal number of central memory words) that the job will require.

When the CM parameter is specified, that amount of storage is allocated to the job throughout execution, unless the job itself requests a smaller amount by a REDUCE or RFL card. If the CM parameter is not used, the system will establish field length requirements for each step of the job, expanding or contracting it as necessary. Since smaller field lengths are used whenever possible, more jobs may pass through the system in a given time period.

The SCOPE library programs, including the loader, compilers, and utilities, have an associated field length in the library tables. The field lengths will be set by the installation to a judicious length for typical jobs, which should eliminate the need for the CM parameter on many job cards.

Any CM parameter on the job card will be rounded upward to a multiple of 100 by SCOPE. The highest permissible value is defined by the installation. An RFL control card requesting a field length greater than the CM value on the job card will cause job termination. The RFL limit is the installation field length maximum if CM is not on the job card.

A typical job card for a COMPASS assembly job may be:

```
ASSEM,T60,CM55000.
```

ECfl

fl is the maximum amount (octal) of direct access ECS the job will need, in multiples of 1000-word blocks. Generally, this parameter is specified only for jobs that require large amounts of data. The highest value permitted is defined by the installation. An installation default amount (typically zero) will be assigned if the parameter is omitted and subsequent MEMORY requests will not be allowed to exceed that amount.

The EC parameter is not required when mass storage files are buffered through ECS (as explained with the REQUEST card).

This job would be given 4000 (octal) words of ECS:

```
TODAY,CM43000,EC4.
```

Pp

p is the priority level (octal) requested for a job. The lowest executable priority level is 1. If zero is given for p, the system will treat it as level 1. The installation determines the highest value permitted, but it never can exceed octal 7777. A value greater than the highest permitted value will default to the installation default. A job of very low priority, might have a job card as follows:

```
JOB,CM500000,P1.
```

Dym

This parameter is used only in conjunction with a string of interdependent jobs. y is the dependency identifier (two alphabetic characters) assigned by the user to the entire string. m is the dependency count (number) of jobs (0-77 octal) upon which this particular job depends. Examples using the D parameter are presented in the discussion of the TRANSF card.

A job that needs the results of two previous jobs for execution may have a job card:

```
JBTHR,CM60000,DXX2.
```

MTk
NTk

One of these parameters must be selected if SCOPE, rather than the operator, schedules tape unit use. The installation determines whether this parameter is necessary. This parameter is used by the system for tape unit scheduling purpose only. It does not change requirements for a REQUEST for a physical unit.

TPk

MT specifies 7-track tape and NT 9-track. If both 7- and 9-track tapes are used, MTk and NTk should both be noted. TP has the same meaning as MT. But if both TP and MT appear on the job card, only the MT parameter will be used to determine the maximum 7-track tape units needed.

k is the maximum number tape units a job will require at any one time. k can range from 0 to 77 (octal), but cannot exceed the total number of tape units at the computer site. If more tape units are required at any time during job execution than are specified by k, the job will be terminated.

A job can use more than a total k tape units as long as their use is not simultaneous. For example, if k is 3 and 7-track tape units A, B, and C are assigned to the job, and an UNLOAD, but not a RETURN function is issued for the tape unit C, tape unit D can be requested for the job. This makes a total of 4 tape units used during the entire job. For this job the card might appear as:

```
TAPEJOB,MT3.
```

CPp

This optional parameter is applicable primarily to dual processor installations. Each dual-processor system has a primary and a secondary processor. Also, the 6000 can operate as a station in a 7600 system; in this case it assigns jobs for execution on either the 7600 or the 6000 as specified by this parameter. The value of p can be determined from the following chart:

System Configuration	Processor to Be Used				Comments
	Primary	Secondary-ary	7600 only	7600 6000	
6400 (CDC CYBER 70/Mod 73-1X) Single 6400 Processor	*	*	70	76	See 7000 SCOPE 2 RM
6500 (CDC CYBER 70/Mod 73-2X) Dual 6400 Processors	A	B	70	76	A/B used for CE diagnostics
6600 (CDC CYBER 70/Mod 74-1X) Single 6600 Processor	*	*	70	76	See 7000 SCOPE 2 RM
6700 (CDC CYBER 70/Mod 74-2X) Dual 6600/6400 Processor	66	64	70	76	A/B may be used

*When SCOPE 3.4 is running on a single processor 6400 or 6600 system, these parameters have no meaning and should not be used.

When this parameter is omitted on jobs executing on a dual-processor system, the job uses whichever central processor is available.

This example specifies use of the 6600 central processor.

```
M2,CM43000,CP66.
```

After the terminator following the last parameter, general comments or installation defined material can appear on the job card.

Examples of job cards:

```
J2,T4,CM45000,EC2,P1,DAB3,MT5,CPA.  THIS IS A VERSION 3.4 JOB.

TO23(EC1(MT1(CM43000)

START(T3)

THIS JOB CARD GIVES ALL DEFAULT VALUES AND JOB NAME THISJ.

OSCAR.COBOL V3 USED

S3R2,MT1. FIRST RUN.

DOGCAT,CM50000.
```

LIMITING MASS STORAGE (LIMIT Card)

Normally, a job is assigned as much mass storage as it needs; however, a user may want to limit the maximum mass storage that should be assigned, for example, during a debug phase when large amounts of output would indicate program errors. With the LIMIT card, the user restricts mass storage allocation. Any time mass storage in excess of this limit is required, the job terminates.

```
LIMIT(n)
```

n Octal number indicating the multiple of 10000 octal 60-bit words that should be allocated to the job

The value of the LIMIT parameter should anticipate both the number and size of files that will exist at one time. Generally, very small limits should be avoided, since the system allocation of one record block, at minimum, for each file may exceed the limit established even though each file is small.

Record blocks are defined at each installation, usually with different sizes of blocks for different mass storage devices. A 6638 disk, for example, may have record blocks of 6200 octal words. A control card specifying LIMIT(2) would, in this instance, cause job termination when a third file is opened, since 3 times the record block size is more than the stated limit of 20,000 words.

Mass storage occupied by the INPUT file or attached permanent files is not involved in the total mass storage allocation for LIMIT card calculations. Any file evicted from mass storage decreases the count of words allocated.

CHANGING CENTRAL MEMORY FIELD LENGTH

The REDUCE and RFL cards affect the central memory field length assigned to a job. REDUCE causes a smaller field length to be assigned; RFL may be used to increase or decrease field length.

Since, for many programs, memory requirements differ significantly from those required for compilation or load, large field lengths are not necessary for the entire job. System utilization can be improved when excess field length is freed for other jobs. Under SCOPE 3.4, the system dynamically adjusts field length requirements only when a CM requirement is omitted from the job card. If a job will need field length larger than would be assigned for any product set member used in the job, the CM parameter for the maximum field length required should be specified on the job card. The user then can specify REDUCE/RFL requests as needed in the job.

The memory available when the system increases user field length in response to a compiler or assembler call will be determined by the job card specification.

REDUCING FIELD LENGTH AFTER LOAD (REDUCE Card)

The REDUCE card is used preceding cards that execute a program. Central memory field length will be decreased to the amount of memory specified to execute the loaded program.

```
LOAD(ABLE)
REDUCE.
EXECUTE.
```

NEW FIELD LENGTH REQUEST (RFL Card)

With the RFL card, the user can request a different central memory field length during job execution. Field length can be made larger or smaller.

```
RFL (fl)
```

fl New field length in octal (decimal by installation option)

The maximum field length that can be requested is the value for the CM parameter on the job card. If CM is not used on the job card, the maximum available, based on an installation parameter setting, establishes the maximum limit for an RFL.

SETTING PROGRAM SWITCHES (SWITCH Card)

In program branching, where two alternate processing routes are provided, the software sense switch is frequently used to determine the path taken. This switch is a bit in central memory that a user's program can reference. A program might contain a request to take one path if the bit is set to one (on) and another if it is zero (off).

Up to six switches can be set or reset by SWITCH cards in the control card record of a source program. At the start of every job, all switches are zero.

```
SWITCH(n)
```

The parameter n must indicate the number (1-6) of the switch to be manipulated. The following control card would turn on switch 4.

```
SWITCH,4.
```

A switch can be reset to zero prior to job termination by including a second SWITCH card referencing the same switch, as shown in this example.

SWITCH,4.	Set switch to 1.
SWITCH,4.	Resets switch to 0.
SWITCH,4.	Resets switch to 1.

Switch reference instructions vary, depending on the program source language. They are described in the reference manuals covering each language.

COMMENTS ON CONTROL CARDS

Informal comments or remarks can be inserted after the terminator on any control card. Comments appear in the printed job dayfile, and also in the dayfile display for the operator on the console screens. Examples follow:

```
JOBSAM,T500,CM50000,MT1.  WORK ORDER NO. 2126A.
REQUEST,TAPE1,MT.
FTN.                      THIS IS NEW COMPILER
LGO.
7/8/9
```

INSERTING FORMAL COMMENTS (COMMENT Card)

Comments can be inserted independently of other control card requests (except between two or more loader control cards) by punching them on COMMENT cards, in this format:

```
COMMENT.n . . . . n
```

Comments or remarks (n . . . . n) are inserted after the period following the word COMMENT. They may be up to 72 characters and occupy any column, 9 through 80. Any character can be used, including blanks, colons, commas, periods, and special characters. COMMENT cards are printed in the job dayfile and displayed to the operator on the console screen. Only the characters in columns 9 through 80 appear, however, and not all special characters display. All characters will be printed on the output.

Since the COMMENT card requires no action by the computer operator, and job processing does not halt for pending action, the computer operator may not notice all messages because of the speed at which this card is processed. It is better to place operator instruction after the parameter list of cards such as REQUEST, LABEL, or RPACK which require operator action.

If a comment is too long for one card, it can be continued on as many COMMENT cards as necessary, as shown in this example:

```
job card.
FTN.
COMMENT. THIS JOB CALCULATES THE SPECIFIC IMPULSE DERIVED FROM THE
COMMENT. E-G INJECTOR, UNBAFFLED VERSION; USED WITH GAS
COMMENT. GENERATOR 11117A, ON THRUST CHAMBER YLR2776A
LOAD(LGO).
REDUCE.
EXECUTE.
7/8/9
```

REQUESTING EQUIPMENT FOR A JOB

Before a file can be referenced by a job, the device on which it resides or is to be written must be identified. For files not on public devices, the device must be assigned to the job.

A public device is a mass storage device used by the system to hold system files, permanent files, and default named files such as INPUT and OUTPUT. All public devices are allocatable – they can be shared by more than one job at the same time. If a programmer does not specify otherwise, his files will be assigned to a public device.

A private device is a mass storage device that holds files by specific request. The two private devices, family packs and sequential packs, are non-allocatable devices that can be assigned to only one job at a given time.

For files input from punched cards or output to a printer or card punch, SCOPE automatically makes the assignment. All card input is written on file INPUT, printer output on file OUTPUT, and card output on PUNCH (for Hollerith cards) or PUNCHB (for binary cards); all are files on public devices. When card input is referenced by a job, INPUT is read automatically. When the job is terminated, OUTPUT is printed, PUNCH is punched in Hollerith format, and PUNCHB is punched in binary format.

Automatic public device assignment results for all files created while a job is in progress unless the user specifies another device. Thus, if a job created a file named SAMMY and did not specify a storage device. SAMMY would be written on a public device.

For input files from devices other than the card reader, or for output devices other than public devices. card punch or printer, the user must request equipment with a REQUEST card (LABEL card for some tape files; RPACK for family pack files). Since control cards are processed in order of appearance, the REQUEST card for a particular file must precede the control card that executes the program referencing that file. Otherwise, the file will be sought or written on a public device when it is referenced.

REQUEST cards are most commonly used with magnetic tapes and sequential packs, but they can be used to cause file assignment to any public device or unit record equipment. Files are assigned to public disk packs with a REQUEST card or by system default. For family disk packs, however. these private devices must be assigned to the job with an RPACK control card before a REQUEST card assigns a file to the pack.

When a REQUEST card is encountered, job processing may halt for operator action or continue with SCOPE action, depending on the form of the parameter specifying device type and. for magnetic tape. the installation tape assigning options.

An asterisk preceding the device type mnemonic causes SCOPE to assign the device without operator action. The tape assigning options available with SCOPE 3.4 make the * redundant for magnetic tape requests, but it may be used; however, * cannot be used if two units are requested with the same card. or a multi-file set is involved. If * is not used for devices other than tape. the REQUEST card will appear on the operator display for a manual assignment. The operator must then make the unit physically ready and logically assign it to the job by entering a command on the console keyboard. Processing resumes and the device is recognized as the source or destination of the file.

General form of a REQUEST card:

```
REQUEST(lfn,dt,eq)    or    REQUEST(lfn,*dt,eq)
```

lfn — Logical file name by which file will be known throughout the job. This parameter is required and must be the first parameter; other parameters are optional and may appear in any order.

lfn must be 1-7 letters or numbers, the first of which must be a letter.

dt — Device type mnemonic plus other dt parameters to further describe equipment requested. An asterisk allows assignment of devices without operator action if possible.

eq — Equipment status table ordinal of device; to be used only under controlled circumstances.

The optional device type descriptors depend on the category of equipment involved. Details of parameters for the REQUEST card are discussed separately in relation to files on the following devices:

Unit record devices such as card reader and line printer

Public devices including those used for permanent files

Magnetic tapes (7- and 9-track) including multi-file sets

Private devices called family packs and sequential packs

The eq parameter is most useful when REQUEST is entered at the console by the operator or the programmer. It is not recommended for use otherwise.

When sufficient information is given in the REQUEST card, SCOPE will assign the device to the job without operator action. Device eq will be assigned if it is available. If eq is not on the control card, dt will be examined. An asterisk prefixing the dt parameter causes SCOPE to assign the device automatically. For tape requests, a VSN parameter is used to locate and assign the tape if it is mounted. For other device requests, operator action is required if an asterisk does not precede the dt parameter. If neither dt nor eq is declared, the operator may assign any device.

SCOPE compares the device assigned by the operator with the request; any discrepancy is reported to the operator. An additional operator command must be given if the dt parameter on the control card is to be overridden by manual assignment. Conflicts between dt and eq parameters also must be resolved by the operator.

REQUESTING UNIT RECORD DEVICES

When a file is input from a card reader or output to a printer or card punch, devices are assigned automatically by SCOPE; the REQUEST card is not necessary. However, if a special type or model of card reader, card punch, or line printer is to be used, this device must be requested.

The REQUEST card to associate files with unit record devices is written:

```
REQUEST(lfn,dt)    or   REQUEST(lfn,*dt)
```

The lfn and * have the same meaning as explained with the general REQUEST card format.

Device type mnemonics, dt, for unit record equipment:

LP	501 or 512 line printer
L1	501 line printer
LQ	512 line printer
CR	405 card reader
CP	415 card punch

The following device types are recognized, but not supported by the standard SCOPE system. If an installation provides software drivers for these devices, they may be specified on the REQUEST card:

GC	252-2 graphic console
HC	253-2 hard copy recorder
FM	254-2 microfilm recorder
TR	Paper tape reader
TP	Paper tape punch

REQUESTING PUBLIC DEVICES

Any file created without a request for a specific device is assigned to a public device by default. A REQUEST for a public device should not be used unless the programmer needs a particular device. However, if the file is to become a permanent file, a REQUEST card for a permanent file device is required.

REQUEST card to associate files with public devices is written:

```
REQUEST(lfn,dtaa,OV,ECn)    or   REQUEST(lfn,*dtaa,OV,ECn)
```

The lfn and * have the same meaning as in the general format explanation.

Device type, dt, mnemonics for public devices:

AA	6603 disk
AB	6638 disk
AC	6603-II disk
AL	821 data file
AM	841 multiple disk drive
AP	854 disk pack device
AF	814 disk file
AD	865 drum
AX	ECS
A*	Any mass storage device
PF	Any permanent file device

When ECS residence is specified by dt AX, the system will assign the file automatically; *AX is not necessary.

Allocation style aa is an optional appendage to the device type mnemonic. The two-digit octal codes representing allocation style must be defined at each installation; they can be used to identify sub-areas of a device. Files then can be assigned to specific sub-areas of a device by including the appropriate allocation style in the request.

The OV parameter allows a file to be written or to overflow to another device if the mass storage indicated by the dtaa parameter is not available. The system then will assign any mass storage; however, files assigned to permanent file devices will not be written to non-permanent file devices. (If all mass storage of any type becomes unavailable during writing, a device capacity exceeded status will be returned to a COMPASS program if the EP bit is set in the file environment table.)

The EC parameter can be used with any public device or with family pack files, to have the file buffered through ECS between the device and central memory. An installation can make provision to have all sequentially accessed files buffered through ECS without the need for this parameter.

EC	Default buffer size used
ECnnnn	Buffer nnnn words multiplied by 1000 octal
ECnnnnK	Same as ECnnnn
ECnnnnP	Buffer nnnn ECS pages

Default buffer size and page size are defined by the installation. The system uses pages as units of allocation for ECS.

Input/output buffering through ECS is valuable mainly for sequential files that are not repositioned frequently. The ECS buffer should be at least the same size as the largest record block defined on any mass storage device in the system. There is no reason to have a large circular buffer in central memory; 201 to 401 (octal) words are adequate, since the transfer rate is primarily a function of the size of the ECS buffer. The job card need not specify ECS requirements if ECS is used only for file buffering.

The programmer uses the ECS buffering feature with mass storage read or write operations. On a write function, system programs will transfer data from the file circular buffer in central memory to the ECS buffer. The ECS buffer grows, as it is filled, to the maximum size defined on the REQUEST card. The full buffer then is written to mass storage. On a read, the ECS buffer is filled immediately from disk, with data transferred to the circular buffer in central memory as the circular buffer is emptied.

If device type AX for ECS residence is used with the EC parameter, that parameter will define the maximum size of the file. Any file overflow will be handled according to the setting of the EP bit in the FET, with the job terminating or returning control to the user.

In this example, the file will be buffered between central memory and a default public device:

```
REQUEST(BIGFILE,EC)
```

The following examples relate FILE1 to any 6638 disk available:

```
REQUEST,FILE1,AB.          operator must assign file to disk
REQUEST(FILE1,*AB)         operator action is not required
```

FAMILY PACK PROCESSING

A family pack (formerly identified as a private pack) is mounted on a unit for exclusive use by a single job and removed after the job terminates. In use, the device is non-allocatable. Files on the pack can be written in sequential or random format.

As many as 63 (decimal) files can exist withın a famıly ot packs. If one pack cannot hold aii fiies, the system will instruct the operator to mount other packs as needed, up to a maximum of five packs.

Two control cards are required before a file on a family pack can be written:

RPACK control card to assign the pack to the job

REQUEST card to associate the file with the pack

To read existing files, only an RPACK card is needed to assign the pack to the job. To remove files from family packs, a REMOVE card is needed.

The RPACK card gives the pack a name that must be used subsequently on the REQUEST cards for pack files. This name is associated with the entire pack, and with the family of packs that may be created if files overflow to additional packs. The pack name and the volume serial number together produce unique identification of any pack.

The REQUEST card defining a logical file name must specify the pack on which it is to be written. Once a file exists on a pack, further REQUEST cards are not needed to access the file in future jobs. All information defining the file and its location on the pack will be preserved on the label during termination of the job that creates the file. The RPACK card with an E parameter will make files on that device available to the system the next time the pack is mounted. They can be read, modified, or deleted by any job that contains the RPACK pack identification.

ASSIGNING FAMILY PACK TO JOB (RPACK Card)

The RPACK control card that requests assignment of a family pack must specify whether the pack can be considered empty with no files or an existing pack with files to be preserved.

To request assignment of a new family pack to a job:

```
RPACK(pname,N)    or   RPACK(pname)
```

To request assignment of a pack with existing files:

```
RPACK(pname,E)    or   RPACK(pname,E,vsn)
```

The pname parameter is the pack name to be assigned to the entire pack or resulting family of packs; it is not to be confused with the logical file name associated with each file to be written on the pack. However, the pack name can be the same as the name of one of its files. If neither N nor E is specified, N is assumed.

The vid parameter is a visual identifier number corresponding to the identification on the outside of the pack and, in packs with existing files, to a field in the pack label. The vid parameter must be 1-6 characters with no restrictions for an initial letter; vid is optional; if it is not used, a comment with this number should appear on the card to guide the operator in assigning the pack. The system will ensure that the operator assigns a pack with a corresponding vid in the label.

The vsn of the pack assigned for a new family is reported in the job dayfile. For packs with existing files, the system will ensure that the operator assigns the correct pack only if the vsn parameter is specified on RPACK. The system will not complete assignment until the correct pack has been mounted and then the system verifies that the pack family name is correct.

Only one RPACK control card per family should be used regardless of the number of physical packs in the family. When this card is processed, SCOPE will instruct the operator to mount all packs in the family.

REQUESTING FAMILY PACK FILES

Each file to be written on a family pack must be identified by a REQUEST card in the format:

```
REQUEST(lfn,PK,pname)
```

The lfn parameter is the logical file name. PK and pname parameters are required. They result in file lfn being assigned to family pack pname without further operator action. An * prefix is not needed for this automatic assignment, and it must not be used.

DELETING FILES FROM FAMILY PACKS (REMOVE Card)

A file can be removed from a family pack by first issuing the RPACK card to assign the pack to the job and then issuing REMOVE. When the REMOVE card is processed, all space on the pack occupied by the named file is released, and all system references to the file are destroyed. No operator action is required.

REMOVE card format:

```
REMOVE(lfn,pname)
```

The name of the file to be removed, lfn, is required.

The pname parameter, for pack name, is optional; when it is used, SCOPE compares pname on the REMOVE card with pname on the label on the pack. If pname is the same on both, the file will be removed; if it is not, a message is issued to the operator.

The following two jobs create, access, and delete files on a family pack. In JOB2, no REQUEST card is needed for FILE1 because it already resides on the pack and is known to the system.

```
JOB1.
RPACK(MYPACK,N)                              Requests blank labeled family pack
REQUEST(FILE1,PK,MYPACK)
REQUEST(FILE2,PK,MYPACK)
RUN(S)                                       Compiles FORTRAN program
LGO.                                         Executes program
7/8/9
FORTRAN program to write files FILE1 and FILE2
6/7/8/9

JOB2.
RPACK(MYPACK,E,1PACK)
COMMENT.  RPACK CARD RESULTS IN OPERATOR ASSIGNMENT
COMMENT.  OF PACK WITH VSN 1PACK MAKING FILES FILE1 AND
COMMENT.  FILE2 AVAILABLE TO THE JOB.
REQUEST(FILE3,MYPACK,PK)                     Assigns new file to pack
RUN(S)
LGO.
REMOVE(FILE2)                                Deletes FILE2 from pack
7/8/9
FORTRAN program that uses FILE1 to create FILE3
6/7/8/9
```

SEQUENTIAL PACK PROCESSING

A sequential pack contains a single sequential format file for use by a single job. No file size limit is imposed; the file can extend over many packs in the same way that a magnetic tape file can extend over many volumes. Sequential pack files, unlike family packs, are not limited to the number of units available, since only the portion of the file currently in process need be mounted. These packs can be dismounted during job execution when they are not in use.

Sequential packs are requested for use by the REQUEST card. The DP parameter indicates a sequential pack. A new pack is requested by:

```
REQUEST(lfn,DP,VID=nnnnnn,dt,N)    or    REQUEST(lfn,2DP,VID=nnnnnn,dt,N)
```

lfn — Logical file name

DP — Sequential pack identifier. If 2 precedes DP, the operator will assume a multi-pack file is involved and assign 2 units. The two assigned units must be the same device type.

nnnnnn — Visual identifier of 1-6 characters. The VID must refer to the first pack of the sequential set. Adherence to COBOL/ANSI standards for access of subsequential packs is handled by the compiler.

dt — Device type mnemonic:

AP	854 disk pack
AM	841 multiple disk drive

N — New pack indicator. Default is N.

Parameters after lfn are order independent; only lfn and DP are required. If VID is not supplied, the operator will supply it when the pack is mounted. The VID will be incorporated in the label of the pack to verify that packs assigned are those requested. Each pack will have a unique VID; the VID of the first pack of a multi-pack file will be used to identify all packs in that multi-pack file as well.

An existing pack is requested with:

```
REQUEST(lfn,DP,E,VID=nnnnnn,dt)
```

All parameters except dt are required for existing packs; they may be in any order.

When a file spans more than one pack, two pack drives can be requested with 2DP. Processing can continue on the second pack while the next pack is mounted on the first unit. The first unit is turned off when its pack is processed and must be turned on when the next pack is mounted. If multiple packs are involved, the operator is responsible for the order in which they are accessed. They must be accessed sequentially starting with the first in the series. If they are mounted out of order, the system will display a message and give the operator the option to continue processing or to mount another disk pack and recheck the label.

Operator action always is required to assign sequential pack units to the job. An * should not be used with the DP parameter.

Multipack files can be processed on one or two units by programmer request. During file processing, the operator is instructed to mount subsequent packs as necessary.

Examples:

The following cards request new packs:

`REQUEST(SEQFILE,DP)`	Default N; operator will supply VID
`REQUEST(SFILE,VID=12345,N,DP)`	Operator must mount pack having VID. 12345
`REQUEST(SEQUENT,2DP,N,AP)`	Operator must assign two 854 packs

Packs with existing files are requested in the following examples:

`REQUEST(SEQ,E,VID=222222,DP)`	System will verify operator assignment of pack with VID 222222
`REQUEST(SEQTL,2DP,E,VID=122221)`	First pack that operator assigns must have VID of 122221; system will request additional packs in file by VID.

MAGNETIC TAPE PROCESSING

Magnetic tape files to be used or created by a job must be explicity requested. Three control cards may be involved: REQUEST, LABEL, and VSN.

The REQUEST card can be used for all tape files – labeled, unlabeled, single file, or multi-file set. Parameters, in addition to specifying format and density, can specify processing for the file. Identifying the tape as input or output and the type of label is sufficient to initiate label processing and checking when the file is opened. The installation default options for unloading, label processing, and parity error processing may be overridden. A volume serial number parameter for the reel (or first reel in multi-reel file) will allow the system to assign the file automatically.

The LABEL card can be used in place of a REQUEST card for a labeled, single file reel and to write or check file header labels on single or multi-file reels. Parameters on the card establish label type and whether labels are to be read or written. Fields in file header (HDR1) labels will be written or checked according to the values on the card. If a multi-file reel is to be labeled, a REQUEST card must first establish the multi-file name, then a LABEL card may exist with the name and label field values for each file in the set. With the LABEL card, either a volume serial number or a label name may be given for identification for automatic tape assignment purposes. Automatic assignment by label name applies only when the read (R) parameter is specified on the LABEL card. The LABEL card also can be used to position to a particular member of a multi-file set.

The VSN card may be used to equate a file name with a volume serial number so that the system can assign a mounted tape automatically when it is requested by a REQUEST or LABEL card or function. The VSN for a multi-file set or for alternate reels can be stated. Since the system accepts the first VSN equated to a file name, a VSN card preceding a REQUEST or LABEL card overrides any VSN value on those cards or supplies the omitted parameter. This VSN information is independent of label information. It is not written or checked against label fields.

The automatic tape assigning features of SCOPE 3.4 (selectable by installation options) speed job throughput when the programmer supplies information to allow assignment of mounted tapes without operator action. The system will search first for an eq parameter, then a VSN parameter, then a label name from among the control cards. If both the VSN and label name parameters are specified, only the VSN is used for automatic assignment. However, label verification proceeds separately and inconsistencies will be brought to the attention of the operator for action. The operator has the option of assigning a VSN to a tape when it enters the system if such identification was not made by the programmer.

REQUESTING MAGNETIC TAPE FILES (REQUEST Card)

The REQUEST card can describe both physical and logical characteristics for magnetic tape files. When only the logical file name and magnetic tape device type MT are specified, the file, by default, becomes a 7-track, unlabeled tape with SCOPE standard records written at installation density, or read at written density; and installation declarations for automatic unloading are honored. Any other use, such as for checkpoints or multi-file sets, or characteristics of the file must be specifically declared on the REQUEST card.

Format of REQUEST card:

```
   REQUEST(lfn,dt, . . . )
```

The logical file name and a 7- or 9-track device type mnemonic is recommended. Further definition of the file is made by adding dt parameters. If the request involves a multi-file set, the MF parameter must appear, the first parameter will then be the multi-file set name (mfn). Also, mfn may not be used in any I/O request except as the M parameter in LABEL or POSMF request.

The MT or NT device type parameter may be prefixed by an asterisk or a 2. The asterisk is applicable only when compatibility with previous operating system is considered. The asterisk prefix results in assignment of a scratch tape to the file. However, if a non-scratch VSN has been specified also, it will override the scratch designation. If the REQUEST card includes parameter E, a scratch tape will not be assigned. Depending upon the selection of installation options, SCOPE will attempt to assign the tape to a job automatically using an eq, VSN, or labelname parameter. Operator assignment is necessary only when automatic assignment attempts are unsuccessful.

If either a 7- or 9-track tape is acceptable, an MN parameter can be used in place of MT or NT or a density parameter which implies MT or NT. The resulting tape will have default density.

A 2 prefix to MT or NT will cause two tape units to be requested from the operator; they will be used in the order assigned. Tape requests using the 2 prefix cannot be auto-assigned. When the tape on the first unit reaches end-of-reel, the system begins processing the tape on the second unit while the tape on the first unit is rewound and unloaded. When the tape on the second unit reaches end-of-reel, the system returns to the first unit, which should have been mounted in the interim with a new tape. The tape on the second unit is rewound and unloaded. This alternating process is repeated as long as the file is referenced. The operator must ensure the proper tape mounting sequence.

Characteristics of the tape may be declared by additional parameters as shown below. No more than one mnemonic for each parameter in braces may be used. Parameters are not order dependent.

7-TRACK TAPE PARAMETERS:

$$\texttt{MT,}\ \left\{\begin{matrix}\texttt{LO}\\\texttt{HI}\\\texttt{HY}\end{matrix}\right\},\ \left\{\begin{matrix}\texttt{CK}\\\texttt{MF}\end{matrix}\right\},\ \left\{\begin{matrix}\texttt{S}\\\texttt{L}\end{matrix}\right\},\ \left\{\begin{matrix}\texttt{U}\\\texttt{Y}\\\texttt{Z}\end{matrix}\right\},\ \left\{\begin{matrix}\texttt{E}\\\texttt{N}\\\texttt{NS}\end{matrix}\right\},\ \left\{\begin{matrix}\texttt{IU}\\\texttt{SV}\end{matrix}\right\},\quad \texttt{NR}\quad ,\quad \texttt{VSN=uuuuuu}$$

7-track identification:

A declaration of LO, HI, or HY is sufficient to define the device type as MT. If MT is absent, LO, HI or HY may be prefixed by a 2 if two units are required.

Density:

LO 200 bpi density

HI 556 bpi density

HY 800 bpi density

absent Density will be set to an installation defined value if initial use is output. If initial use of a labeled tape is input, the density of the label will be determined automatically; however it is recommended that density be specified whenever known, and that density will be used to read both the label and the data, except as indicated under Z below. If initial use of an unlabeled tape is input, the density will be set to an installation declared value.

File disposition:

IU Any physical unload of the tape file in a context other than reel swapping will be inhibited. The IU parameter does not inhibit logical actions associated with UNLOAD or RETURN. IU is recommended when a scratch tape or input tape is requested that is to remain mounted and ready.

SV The tape file will be unloaded at job termination, and the operator will be notified that the tape is to be saved.

absent Action performed at end of job is option of the installation.

Volume serial number identification:

VSN=uuuuuu The VSN parameter designates the 1-6 character volume serial number of the tape reel. The VSN will appear on the previewing display for the operator's information before the job is assigned to a control point. SCOPE will use the VSN to locate the tape reel. Once the tape is assigned, the VSN will be verified against the standard or Z format label, if present. VSN also will be verified against operator-supplied VSN for an unlabeled tape.

If a scratch tape is desired, a VSN of SCRATCH or 0 can be used. The * prefix may be used for a scratch tape also.

If a VSN is declared for a file on a REQUEST and a VSN card or on a VSN and a LABEL card, the first declaration will be effective.

absent The VSN card declaration will be used if present; otherwise, file header label fields will be used for assignment and verification. If neither VSN nor file header label field declaration is made, any tape reel will be accepted; but the assignment must be made manually unless * prefix is used.

Parity error recovery procedure:

NR The NR parameter may be used to inhibit normal parity error recovery procedures. Data containing the parity error will be returned to the user.

Special tape use:

CK Checkpoint dumps will be written on the tape.

MF The tape is a valid U or Z labeled multi-file set.

absent Neither of the above is assumed.

Data format:

S Data format is S.

L Data format is L.

absent Data format is SCOPE standard.

Input or output use (apply only to labeled tapes):

E Existing label. Initial use of the tape is input; tape label will be read and checked.

N New label. Initial use of the tape is output; tape label will be written.

absent If file is to be labeled (U, Z or Y is declared), a tape label will be written.

Label characteristics:

U Tape label format is ANSI. (SCOPE standard label).

Y Tape label format is Y. (3000 series label)

Z Tape label format is ANSI, except character 12 of the VOL1 label is used to indicate data density. These labels were standard for SCOPE 3.3.

absent Tape is unlabeled unless either E or N is declared; in which case, ANSI (U) label format is assumed.

Label processing:

NS The NS parameter may be used to indicate a tape has non-standard labels and is to be processed as unlabeled even though the tape is labeled; existing labels will appear to the system as data. Either the labels are to be ignored or the user will process them.

9-TRACK TAPE PARAMETERS:

A declaration of NT or a 9-track density for a tape to be written is required to identify a 9-track tape. Definitions and conditions for all except the density and data format parameters are the same as those for 7-track tape.

$$\mathtt{NT,}\ \begin{Bmatrix}\mathtt{PE}\\ \mathtt{HD}\\ \ \end{Bmatrix}\mathtt{,S,}\begin{Bmatrix}\mathtt{CK}\\ \mathtt{MF}\\ \ \end{Bmatrix}\mathtt{,}\ \begin{Bmatrix}\mathtt{U}\\ \mathtt{Y}\\ \mathtt{Z}\end{Bmatrix}\mathtt{,}\ \begin{Bmatrix}\mathtt{E}\\ \mathtt{N}\\ \mathtt{NS}\end{Bmatrix}\mathtt{,}\ \begin{Bmatrix}\mathtt{US}\\ \mathtt{EB}\\ \ \end{Bmatrix}\mathtt{,}\ \begin{Bmatrix}\mathtt{IU}\\ \mathtt{SV}\\ \ \end{Bmatrix}\mathtt{,}\quad \mathtt{NR}\quad\mathtt{,}\quad \mathtt{VSN=uuuuuu}$$

Density:

A density specification is effective only when the tape is to be written; density setting is a hardware function when the tape is read.

PE 1600 cpi

HD 800 bpi

absent Tape will be written at installation declared density.

Data format:

S Data format is S

absent Data format is SCOPE standard

Coded data conversion codes for 9-track S tapes: (Refer to conversion tables in Appendix A)

US   Coded data on tape is to be converted from ASCII on input or to ASCII on output.

EB   Coded data on tape is to be converted from EBCDIC on input or to EBCDIC on output.

absent   Coded data conversion is defined by the installation.

REQUEST Card Examples:

```
REQUEST(FILE1,NT,U,E)     or     REQUEST(FILE1,NT,E)
```

The operator must assign an ANSI labeled, 9-track tape. The label will be checked when the first function is issued on the tape. Since density is not specified, it is assumed that both label and data are written at the same density.

```
REQUEST(FILE,*MT)
```

Depending on installation option, the system automatically will assign FILE1 to a scratch tape on a 7-track tape unit. The file will be unlabeled and written in SCOPE standard data format at an installation declared density.

```
REQUEST(STANF27,LO,VSN=OHI017,U,S,SV)
```

Depending on installation option, file STANF27 will be assigned automatically to a unit containing reel OHIO17. An ANSI label will be written; both label and data will be written at 200 bpi. Data format is S. The reel will be saved at job completion.

LABELING TAPES (LABEL Card)

The LABEL control card can be used for the following functions:

Write file header labels in ANSI, 3000 series, or SCOPE 3.3 format

Read and check such labels

Position within a multi-file set

Subsititute for a REQUEST card for a single file reel

Conversion to the use of labeled tape can proceed by using a LABEL(lfn,W,L=z...) card instead of a REQUEST(lfn) card. A labeled input tape can be requested with either the same LABEL card replacing the write (W) parameter with a read (R) parameter or by a REQUEST card with an existing label (E) parameter. Label checking will be performed by a LABEL(lfn,R,....) card. When a REQUEST(lfn,E) card is used, checking ensures only that the tape is labeled. If any of the above checks fail, a message is written to the dayfile; and the operator can mount the correct tape or tell the system to ignore the mismatch and continue.

In most instances, a LABEL card will be the first reference to a file in a job, unless it is preceded by a VSN card indicating the volume serial number of the resident reel. For a single file reel, a REQUEST card is not needed, although a REQUEST card followed by a LABEL card is valid and does not create an error condition. If a REQUEST card follows the LABEL card, duplicate file names are generated; and the job will terminate, since the LABEL program issues a REQUEST function to obtain the equipment. For labeled multi-file reels, a REQUEST card establishing the multi-file set must precede the LABEL cards that write the header labels for various files in the set.

The label program issues an OPEN function to read or write the file label. Contents of the label are copied to both the system and job dayfiles.

A program can inspect file header labels written on a file by issuing an OPEN function. Label information is returned to words 10-13 of the FET for the file or, if the extended label processing features are used, to the file label buffer defined in word 10 of the FET.

Format of LABEL control card:

$$\texttt{LABEL(lfn,}\left\{\begin{matrix}\texttt{W}\\\texttt{R}\end{matrix}\right\},\left\{\begin{matrix}\texttt{Z}\\\texttt{Y}\end{matrix}\right\},\ \texttt{D=d,F=f,N=n,X=x,L=z,V=v,E=e, T=t,C=c,M=m,P=p,VSN=vsn)}$$

The first parameter must be the logical file name; others can appear in any order. The LABEL card can be continued to a second card; if a terminator does not appear on the first card, the next card is assumed to be a continuation of the first.

Default parameters cause a single file header in ANSI format for a file in SCOPE standard format to be processed. Any other label or data format to be written, or a tape to be read, must be declared explicitly.

Read or write parameter: (One must be specified or the job will terminate.)

R Label is to be read and compared with parameters on the LABEL card. When R is used, the tape may be a candidate for auto-assignment by label name.

W Label is to be written.

Label standard:

Y 3000 series label.

Z Label conforms to standard label of previous operating system. Character 12 of the VOL1 label specifies data density; otherwise Z labels are identical to U labels.

absent ANSI label.

Tape characteristics:

d

Density. If omitted, density declared or implied by REQUEST card will prevail.

For 7-track tapes:

LO	200 bpi
HI	556 bpi
HY	800 bpi

For 9-track tapes, the d parameter determines density for writing only; data is always read at the recording density.

HD	800 bpi
PE	1600 bpi

f

Format of file data. Default is standard SCOPE format.

S	S tape format
L	L tape format

n

Code for conversion of 9-track tapes only. Default is installation defined.

US	ASCII code
EB	EBCDIC code

x

Disposition of tape.

IU	Inhibit physical unload
SV	Unload tape at end of job; notify operator to save
CK	Checkpoint dump will be written on tape
CI	Checkpoint dump and inhibit physical unload
CS	Checkpoint dump and save

The 9-track selection can be specified only on the LABEL card by giving either a 9-track density parameter (HD or PE) or a code conversion parameter (US or EB).

File header label fields:

z — Label name: 1-17 characters for ANSI or Z labels; 1-14 characters for Y labels. Default value is spaces.

v — Reel number specifying reel sequence in set. 1-4 digits for ANSI or Z labels; 1-2 digits for Y labels. Default is 0001.

e — Edition number specifying version of file. 1-2 digits. Default is 01.

t — Number of days file is to be retained, 1-3 digits. Default determined by installation. 999 is permanent retention.

c — Creation date, in format of 2 digits for year, 3 digits for date. Default is current date.

m — SCOPE uses this parameter to establish that the current LABEL function applies to a member of a multi-file set; m is the logical multi-file set name as it appears on the REQUEST card for this set, and it must be present for all LABEL cards referencing members of this multi-file set. When the label is written on tape, the multi-file field does not contain the the logical set name; it contains the VSN for the first volume of the multi-file set.

p — Position number indicating file within multi-file set, 1-4 digits. Default is 0001. Not defined for 3000 series labels.

Voiume seriai number:

vsn — Volume serial number of 1-6 characters used to identify the tape for automatic assignment. Parameter may appear on VSN card rather than LABEL card. A VSN of SCRATCH or zero may be used to specify a scratch tape.

AUTOMATIC TAPE IDENTIFICATION (VSN Card)

The VSN card relates the external sticker (tape volume serial number) to the logical file name and also provides information required for the tape job prescheduling display. When this card is used with the REQUEST and/or LABEL control cards or the REQUEST function, it relates a VSN to a logical file name for automatic equipment assignment; otherwise it serves no purpose. A single card, up to the 80-column limit, can contain declarations for more than one file name and VSN. Continuation cards are not allowed, but as many VSN cards as necessary can appear in a control card record.

Format of VSN card:

```
VSN(lfn=vsn, . . . )
```

lfn For a single file, the 1-7 character logical file name

For a multi-file reel, the 1-6 alphabetic character multi-file name

vsn 1-6 character volume serial number.

If any of several alternate reels will suffice, equals signs should separate identifiers, as in: FILE=1234=1235.

VSN cards may be placed anywhere in the control card record as long as they precede the REQUEST or LABEL card for the file named. If a logical file name is to be re-used during a job, such as OLDPL for two UPDATE operations, the first file should be released by an UNLOAD or RETURN control card before a VSN is given for the second file.

Examples:

```
VSN(OLDPL=1234=4567=7890)
LABEL(OLDPL,R,L=SCOPE3P4)
UPDATE.
RETURN(OLDPL)
VSN(OLDPL=0987=7654=4321)
LABEL(OLDPL,R,L=SCOPE3P4REL)
```

The use of the VSN card is recommended in any of the following circumstances:

One or more tape files are requested by the REQUEST function; use of the VSN card allows different tape reels to be used without altering the program.

Multi-reel tape files or alternate reels are used; only the VSN card allows specification of alternate volume serial numbers.

Tape files used by a job deck can change frequently; use of the VSN card allows tape specifications to be changed by replacing a single card.

In the following job, the VSN card has no effect because no REQUEST or LABEL card is specified for file TAPE1. File TAPE1 will be opened as a disk file in this example:

```
JOB5, MT1.
VSN(TAPE1=1234)
REWIND,TAPE1.
```

To have a specific magnetic tape assigned to the job, either of the following requests would suffice:

```
JOB6,MT1.
VSN(TAPE1=1234)
REQUEST(TAPE1,MT,E)


JOB7,MT1.
REQUEST(TAPE1,VSN=1234,MT,E)
```

The maximum number of tape drives a job will use at any time is specified by the MT (7-track) and NT (9-track) tape parameters on the job card. Specifying more tapes than will be needed can delay execution of a job. The greatest delay results from specifying a number of tapes when the job does not use any tapes. Specifying fewer tapes than needed will cause the job to abort.

A poorly set-up job can needlessly tie up system resources and delay its own throughput. For example, consider both jobs JOB3 and JOB4:

```
JOB3,MT3.                                      JOB4,MT1.
VSN(TAPE1=111,TAPE2=222,TAPE3=333)             VSN(TAPE1=111,.......)
REQUEST(TAPE1)                                 REQUEST(DISK,*PF)
REQUEST(TAPE2)                                 REQUEST(TAPE1)
REQUEST(TAPE3)                                 COPY(TAPE1,DISK)
REQUEST(DISK,*PF)                              UNLOAD(TAPE1)
COPY(TAPE1,DISK)                               REQUEST(TAPE2)
COPY(TAPE2,DISK)                               COPY(TAPE2,DISK)
COPY(TAPE3,DISK)                               UNLOAD(TAPE2)
CATALOG(DISK,......)                           REQUEST(TAPE3)
                                               COPY(TAPE3,DISK)
                                               RETURN(TAPE3)
                                               CATALOG(DISK,.......)
```

JOB4 is set up better than JOB3. Although both jobs do the same thing in the same amount of time, JOB3 holds on to tapes that it does not use. JOB3 must wait in the input queue until three tapes become available; JOB4 waits for only one. A good rule is to request tapes only when they are to be used and to release them from the job when no longer needed. Notice that JOB4 uses RETURN instead of UNLOAD; the job will not request another tape for the remainder of its execution.

If conflicting volume serial numbers are given for a single tape file, the first encountered will be used.

Example:

```
VSN(TAPE1=4468)
LABEL(TAPE1,R,VSN=5678)
```

Tape 4468 will be assigned.

If the VSN is absent, 0, or SCRATCH, any available scratch tape may be assigned to the logical file named.

Examples:

`VSN(SHASTA,WHITNEY=0)`	Both files will be scratch tapes
`VSN(SANTA=ROSA/CRUZ)`	2-reel file SANTA begins on ROSA, ends on CRUZ
`VSN(DESERT=MOJAVE=SONORA)`	Either MOJAVE or SONORA can be assigned to file DESERT

LABELED MULTI-FILE SET PROCESSING

A multi-file set consists of one or more files on one or more volumes of tape. Individual files can be accessed by name, even though their order is not known. Multi-file sets are not supported by 3000 series labels.

Labeled multi-file sets require the use of both REQUEST and LABEL cards. (LABEL cards are not required if the program can generate these fields internally.) The REQUEST card specifies the tape characteristics; LABEL produces the file header for individual files. The LABEL card must specify the set name as the M parameter. This set name is limited to six characters and must be different from any local file name. The utility routine, LISTMF, is available to list the labels of all files in an existing set.

The LABEL card itself can be used to position within a set, when a position number is used in the parameter list.

To create a labeled multi-file set, the following parameters should be used (parameters after the first can appear in any order). The label type must be U.

```
REQUEST(mfn,MF,U, . . . )

LABEL(lfn1,M=mfn, W, . . .)

    program call to create lfn1

LABEL(lfn2,M=mfn, W, . . .)

    program call to create lfn2
```

The mfn parameter is the name of the multi-file set, 1-6 letters and numbers beginning with a letter. This parameter associates the file with a particular set: all files in the set must reference it. Also, mfn may not be used in any I/O request except as the M parameter in LABEL or POSMF requests.

On the REQUEST card, the MF parameter designates the first parameter to be a multi-file name rather than a logical file name. The U parameter causes standard labels to be produced. Other parameters on the card should establish tape density and format for the entire multi-file set. On the LABEL card, density and format parameters are ignored. REQUEST can include a VSN parameter.

A LABEL card is recommended for each file. In addition to required lfn and M parameters, optional parameters describing file header fields can appear. If a position number is not given with the P parameter, it is assumed to be one larger than that of the previous file; and the new file will be written at the end of the current set. When an L parameter is used in creating a file header, future jobs can access the file by label name.

To access a labeled multi-file set, a REQUEST card is needed to attach the set to the job. A LABEL card (either U or Z) need appear only for the file to be accessed. For example, to access the third file on a reel:

```
REQUEST(MANY,MF,U, . . . )

LABEL(FILE3,R,M=MANY,P=3, . . . )
```

When an R is specified on a LABEL card, the set is positioned according to the P parameter, an OPEN function is issued to read the label, and the contents are checked against any corresponding parameters on the LABEL card. Use of L instead of P causes the tape to be searched for a matching label name. If a match cannot be found, a message, `FILE NAME NOT IN MULTI-FILE SET,` is issued and processing stops. The same message appears also when neither P nor L is given and the end of the set is encountered. When R is not specified, the next file in the set is opened when P and L are both omitted.

Writing on a multi-file can be done at the end of the existing set; or at some point prior to the end, existing files can be overwritten. For example, to create a new file LASTONE:

```
LABEL(LASTONE,W,M=MYSET,L=LAST)
```

Since P is omitted, the label will be written at the end of existing files and given a position one greater than the last file.

If a position number is given when a label is to be written, the file is positioned as requested. If a label exists at that point, its expiration date is checked. A new label is not written over the existing one unless it is expired or the operator authorizes writing over an unexpired label. Since rewrite-in-place is not defined for tapes, rewriting a file label destroys access to the associated file and all files following it on the tape.

LISTING LABELED MULTI-FILE TAPES (LISTMF Utility)

A list of files in a labeled multi-file set will be printed on file OUTPUT when LISTMF is called. The header label for all files will be listed, even if the set encompasses more than one reel. A REQUEST card defining the set must appear before the LISTMF card.

```
LISTMF(M=mfn,P=p)
```

mfn	Multi-file name same as on REQUEST card
p	1-3 digit position number. Default is 1

The multi-file set mfn is first rewound, then positioned to p. As the content of each file header label is read, it is copied to OUTPUT. Listing continues until the end of set (EOF followed by multiple tape marks) is reached. The reel is not rewound.

AUTOMATIC ASSIGNMENT OF MULTI-FILE TAPES

The assignment of a multi-file tape may proceed automatically with the use of a VSN card under the following conditions:

A VSN card or parameter equates the multi-file name to the physical reel of tape.

```
VSN(mfname=1234)     or     REQUEST(mfn,.....,VSN=1234)
```

A REQUEST card is used to assign the multi-file name to the job.

```
REQUEST(mfname,MF)
```

A LABEL card is used to identify the specific file by label name, equate the file to the logical file name, and identify the file as being a multi-file set member.

```
LABEL(lfn,M=mfname,L=lfn,........)
```

Once the multi-file name has been assigned to the job via the REQUEST card, any file can be accessed individually via the LABEL card. The execution of a new LABEL card automatically prevents the preceding labeled file from being accessed.

DISPOSING OF FILES AND EQUIPMENT

Normally, all files assigned to a job are retained by that job until termination. When the files reside on non-allocatable devices such as magnetic tapes, both the file and the hardware device are unavailable to other portions of the system for the duration of the entire job even though the file is in process for only a short part of the job.

When the DISPOSE, UNLOAD, or RETURN control cards are used, files are released before job termination, making both the logical file name and the resident device available for other uses, within the circumstances noted below. Files named in an UNLOAD or RETURN are unavailable for the remainder of the job. An OPEN function issued later in the job will create another file. The exception involves RETURN of files on family packs, as noted in the family pack discussion.

Permanent files named in either an UNLOAD or RETURN card are no longer available to the job until referenced in a subsequent ATTACH.

RETURNING FILES AND EQUIPMENT

RETURNING EQUIPMENT (RETURN Card)

Any file, including permanent files, can be referenced on a RETURN card to cause SCOPE to perform a CLOSE/RETURN function on the file. All sequential files are rewound; indexes are written for random files. If a file on a public device has a disposition code, it is placed in the output queue; otherwise. its storage space is evicted. The file name is deleted from the system and, except in case of a family pack file. another file with the same name can be created.

RETURN card format:

```
RETURN(lfnl,lfn2, . . . )
```

On this card, lfn is the name of the file released; one or more files can be referenced on a single card.

Magnetic tape output files have trailer labels written before they are rewound. Then they are physically unloaded. With the exception of members of a multi-file set, the tape units on which they reside will be disassociated from the job and made available to the system for new assignment. The count of the number of tape units logically required by the job, as set by a tape parameter on the job card, will be decreased.

Family pack files are locked when the RETURN is executed and become unavailable for the remainder of the job. The file name cannot be reused for another file on the pack.

When RETURN names a file on a sequential pack, the file name is deleted from the system, and the unit is returned to the system for reassignment.

UNLOADING FILES (UNLOAD Card)

SCOPE issues a CLOSE/UNLOAD function for files named in an UNLOAD control card. As with the RETURN card, sequential files are rewound after any necessary labels are written, and indexes are written for random files. File dispositions are honored for files on public devices; otherwise all references to the files are deleted.

Format of UNLOAD card:

```
UNLOAD(lfnl,lfn2, . . . )
```

More than one file can be named on a single card; the file name INPUT may not be named on UNLOAD.

UNLOAD differs from RETURN only in that an UNLOAD referencing a tape file does not affect the count of maximum number of tape units the job requires.

The UNLOAD card cannot override an IU (inhibit unload) parameter on the REQUEST card for the file; if the IU parameter exists, a subsequent UNLOAD will rewind, but not physically unload the tape.

The UNLOAD and RETURN functions differ only in that RETURN reduces the maximum number of tapes that may be held by the job and UNLOAD does not; otherwise they produce the same results.

This job will abort:

```
JOB1, MT1.
REQUEST(TAPE1,MT)
RETURN(TAPE1)
REQUEST(TAPE2,MT)
```

This job will not abort:

```
JOB2,MT1.
REQUEST(TAPE1,MT)
UNLOAD(TAPE1)
REQUEST(TAPE2,MT)
```

DIRECTING OUTPUT BY DISPOSE

Under normal operating conditions, files are released from a job only after all control cards are processed, and the job is released from central memory and terminated. However, a user can request that files on public devices or ECS be released for output processing before job completion. It could be done, for instance, when output is ready for the printer but additional programs in a job remain to be run.

DISPOSE can be used to:

Output files at the central site to specific devices and/or specific forms or cards

Route files to remote sites

Evict files from mass storage

Disposition can take place at the time the DISPOSE card is executed, or it can be delayed until the job terminates. In the control card record, DISPOSE may reference a file before or after it is created. If no previous references to the file exist, DISPOSE will generate one.

To evict a file, removing all system references to file lfn and allowing mass storage space to be overwritten, the format is:

```
DISPOSE(lfn)
```

To output a file to a device at the central site, the format is:

```
DISPOSE(lfn,x)    or    DISPOSE(lfn,*x)
```

The presence of * in the x parameter indicates the file is to be disposed at end-of-job. The * is required if DISPOSE references a file before it is created.

The x parameter represents the disposition requested as indicated by the following codes.

x Code	**Disposition**
PR	Printed on any available printer
P1	Printed on 501 printer
P2	Printed on 512 printer
PE	Printed on 512 printer with 95-character set
PB	Punched on formatted binary cards
PU	Punched on Hollerith cards
P8	Punched on free-form binary cards (using all 80 columns for data)
FR	Printed on microfilm recorder
FL	Plotted on microfilm recorder
PT	Plotted on any available plotter
HR	Printed on hardcopying device
HL	Plotted on hardcopying device

Codes FR, FL, PT, HR and HL are defined but not supported by standard software in SCOPE 3.4.

To output a file to a device containing particular cards or paper forms, the x parameter can be expanded:

```
DISPOSE(lfn,x=Cy)
```

The x parameter is the same as the codes listed above. The y parameter must be two installation-defined. alphanumeric characters identifying a particular card or form. When SCOPE detects the x=Cy parameter, it will tell the operator to insert the necessary forms to carry out the DISPOSE request.

To route a file to a remote site:

```
DISPOSE(lfn,x=Iy)
```

The x parameter is valid disposition code listed above. The y indicates the particular INTERCOM user identification or IMPORT site identification where the file will be output.

The y indicates the particular INTERCOM user identification or IMPORT site identification where the file will be output.

The following job uses DISPOSE. The first program produces a file PAYROLL containing payroll data to be printed on check forms. The second program creates files PUNCHCS and PRINTRM to be punched and printed at different sites.

```
DSPEG.
COBOL.                          Compiles first program
LGO.                            Loads and executes program
DISPOSE(PAYROLL,P2=CCK)         Prints payroll on form CK at central site
REWIND(LGO)                     Rewinds compiler output file
COBOL.                          Compiles second program
DISPOSE(PUNCHCS,*PB)            Punches file at end of job at central site
LGO.                            Loads and executes second program
DISPOSE(PRINTRM,PR=IME)         Prints file at INTERCOM site ME
7/8/9
First COBOL Program
7/8/9
Second COBOL Program
6/7/8/9
```

FILE MANIPULATION

SKIP OPERATIONS

```
SKIPF(lfn,n,lev,m)
```

One or more SCOPE logical records will be bypassed in a forward direction. The request may be initiated at any point in a logical record.

```
SKIPB(lfn,n,lev,m)
```

One or more SCOPE logical records will be bypassed in a reverse direction. The request may be initiated at any point in a logical record.

lfn Logical file name.

n Decimal number of SCOPE logical records, or record groups, to be skippped; default is 1; maximum value is 262,142.

A value equivalent to 262,143 will be treated as a rewind for SKIPB. For SKIPF, a tape file will not be positioned, and a disk file is positioned at end-of-information. Default is 1.

lev Octal number. SCOPE logical records are skipped until the number of end-of-records with level numbers greater than or equal to the requested level is reached; the file is positioned immediately following (for SKIPF) or preceding (for SKIPB) the last record. Default is level 0.

m Mode of file. B for binary files; C for coded files. Default is B.

BACKSPACE SCOPE LOGICAL RECORD

```
BKSP(lfn,n)
```

Multiple SCOPE logical records in the file named lfn are backspaced as specified by the the decimal n. Backspacing terminates if it causes a file to be rewound. Default value for n is 1.

REWIND

```
REWIND(lfnl,lfn2 . . . )
```

lfn is the name of the file to be repositioned. More than one lfn may appear on one REWIND card.

All files specified are rewound.

REWIND positions a file at the beginning of information. For a labeled magnetic tape, this position is the start of the user's data after label information.

In most cases, when a file is requested for a job, that file is positioned automatically at beginning of information. However, because of variations in installation parameters and procedures, automatic positioning may not always occur with every file requested. Therefore, it is best to follow the REQUEST card with a REWIND card to ensure that the. file will be positioned at its beginning when first referenced.

Rewind of a multi-reel file repositions the file at the file beginning; the operator is instructed to mount a prior reel if necessary. Rewind of a member of a multi-file labeled tape positions the tape at the beginning of the specified file, not to the beginning of the reel.

Rewind specifying a multi-file set name is illegal and will cause job termination.

In the following example, the tape containing file MAX is requested for the job; the file is repositioned, loaded into central memory and executed.

```
SEN,MT1.                          Names job.
VSN(MAX=1234)                     Equates MAX with tape 1234
REQUEST,MAX,MT.                   Requests file containing object program MAX.
MAX.                              Loads and executes MAX.
REWIND,MAX.                       Rewinds MAX.
MAX.                              Loads and executes MAX a second time.
7/8/9
data deck
7/8/9
second data deck
6/7/8/9                           Signals EOF and end of job.
```

INTERDEPENDENT JOB PROCESSING

INTERRELATING DEPENDENT JOBS (TRANSF Card)

The user can submit a string of interdependent jobs to the computer, specifying the order in which they are to be executed. In such a string, jobs can be input in any order and from central site or remote card readers. A job will not be executed until all prerequisite jobs in the string have been executed. Whenever possible, SCOPE schedules interdependent jobs for execution in parallel (multiprogramming).

As each job is input, the dependency identifier and dependency count on the job card are noted. The dependency count is decremented by TRANSF cards in prerequisite jobs. When the count of a dependent job becomes zero, it executes.

On the job card, the Dym parameter establishes job interdependency. y is the dependency identifier that names the string to which the job belongs. m is the dependency count (number) of prerequisite jobs on which the job depends.

TRANSF must appear after the control cards that execute the prerequisite programs. The TRANSF card is punched in this format:

```
TRANSF(p1,p2, . . . pn)
```

The p parameter names the jobs for which dependency count is to be decremented. Only the first five characters of each job name are examined by SCOPE, with the dependency string identifier maintaining proper identification. As many job names as will fit on a card can be noted; or multiple TRANSF cards can appear. TRANSF should not appear in the last job in the string since no jobs may depend on it.

An example of an interdependent job string JS follows. Consider jobs with names JOBA through JOBF:

JOBB is dependent on successful execution of JOBA
JOBC on JOBA
JOBD on JOBB and JOBC
JOBE on JOBC
JOBF on JOBB, JOBD, and JOBE

The control card records should appear with:

```
JOBA,DJS00.                JOBB,DJS01.
execution call             execution call
TRANSF(JOBB,JOBC)          TRANSF(JOBD,JOBF)
.                          .
.                          .
.                          .
7/8/9                      7/8/9

JOBC,DJS01.                JOBD,DJS02.
execution call             execution call
TRANSF(JOBD,JOBE)          TRANSF(JOBF)
7/8/9                      7/8/9

JOBE,DJS01.                JOBF,DJS03.
execution call             execution call
TRANSF(JOBF)               7/8/9
7/8/9
```

JOBF, which can execute only if all other jobs in the string are successful, has a dependency count of 3, the number of jobs containing TRANSF references to JOBF.

If a job containing a TRANSF card is terminated before that card is processed, the dependency count of other jobs will not be decreased. Instead, all succeeding jobs that depend on this job will remain in the input queue. No error message indicates that a job in a dependent string has terminated abnormally; operator alertness is needed to know the remaining jobs should be evicted or forced into execution. A message instructing the operator may be placed in a routine after an EXIT card or RECOVR function, but it does not guarantee the operator will see the message.

CONTROLLING TERMINATION PROCEDURES

ESTABLISHING HALT CONDITIONS (MODE Card)

Among the various types of errors that can cause a job to terminate prematurely or branch to an exit path specified by the user, three can be negated so that program processing will continue:

Reference to an operand (any number used in a calculation) that has an infinite value

Reference to an address outside the field length of the job in central memory or ECS; such an address may be generated during assembly if a non-existent location is referenced.

Reference to an operand for floating point arithmetic which has an indefinite value

Normally, these errors will terminate processing; any or all can be suspended as halt conditions, so that processing continues until another type of error is encountered that terminates the job, or until all control cards are executed. The MODE card is used for this purpose.

```
MODE(n)
```

The n parameter is a number specifying the halt conditions to remain in effect for a job:

0 In none of the three negatable cases

1 Only if address is out of range

2 Only if operand is infinite

3 If address is out of range or operand is infinite

4 Only if operand is floating point number of indefinite value

5 If address is out of range or operand is floating point number of indefinite value

6 If operand is infinite or a floating point number of indefinitie value

7 If operand is infinite or a floating point number of indefinite value or address is out of range

The following card will permit processing to continue if a referenced address is out of range of the job field in central memory; processing will halt, however, if an infinite operand or a floating point operand of indefinite value is referenced.

```
MODE,6.
```

Any MODE value that permits processing to continue regardless of a reference to an out-of-range address should be used with great caution. Resulting output probably will have no value. Under such conditions, an attempt to write outside FL appears to complete normally; however, no writing is done. When an attempt is made to read outside FL, zero is returned to the X register specified.

A MODE request remains in effect for a job until a new request is encountered or until the job is completed. At most installations if no MODE card appears in a program, the default is mode 7, which allows a job to halt if any error occurs.

The values on the MODE card are related to the MODE error numbers that appear in the exchange package of the standard error dumps. An additional error number 10, which cannot be negated through a MODE card, indicates a program attempted to use an exchange jump instruction not available in that particular system.

ESTABLISHING EXIT PATHS (EXIT Card)

Normally, when a fatal error occurs which is not a negatable error suspended by a MODE request, processing is terminated, a diagnostic message is issued, and output created prior to the error is output. However, an alternative processing routine can be established within the control card record so that the job can branch in the event of certain kinds of errors. The exit routines will be executed before the job is terminated. Such an exit routine might call for a dump of central memory contents of the job, or it might direct execution of an entirely different program.

Certain conditions cause abrupt termination of a job regardless of an exit sequence:

Request from SCOPE or computer operator to terminate job and inhibit all output (KILL command)

Request from operator to transfer job from central memory back into input queue (RERUN command)

Error on job card

Checksum error during job input

When other types of otherwise fatal errors occur, SCOPE searches the control card record for an EXIT card. The following terminating conditions will result in this search:

Job uses all execution time allotted

Arithmetic error negatable by a MODE card

Peripheral processor encounters improper input/output request

Central processor program requests job termination

Operator request to drop job

ECS parity error occurs

Control card error (other than on job card)

These fatal conditions except the last, can be reprieved within COMPASS, FORTRAN, and FORTRAN Extended programs, as indicated by the RECOVR function discussed in section 12.

If a control card record includes an exit routine but no error occurs, when the EXIT card is encountered the job will terminate as it would if an EOR card had been encountered in the control card record.

The EXIT control card is:

```
EXIT.
```

Errors in control card format, or an attempt to load an object program resulting from erroneous assembly or compilation, result in instant termination of the job even if an EXIT card is present. This action prevents indiscriminate dumping of large loading and compilation routines in cases where an EXIT card is followed by a dump request. To override the instant termination procedure and enter an exit routine regardless of errors, the suffix S should be added to the EXIT card:

```
EXIT(S)
```

The following job illustrates the use of EXIT. Storage will be dumped only if a fatal error occurs to cause the control card processor to search for an EXIT card.

```
MYJOB,P1,T400,CM50000,MT1.              Names job.
REQUEST,MYFILE,MT.                      Requests input tape file MYFILE.
RUN(G)                                  Compiles and executes FORTRAN program.
EXIT.                                   Signals beginning of exit routine.
DMP,1000.                               Dumps first 1000 (octal) words of storage.
7/8/9
FORTRAN RUN program
7/8/9
Data
6/7/8/9
```

PERMANENT FILES 5

INTRODUCTION

A permanent file is a mass storage file cataloged by the system, so that its location and identification are always known to the system. Frequently used programs, subprograms, and data bases are immediately available to requesting jobs without operator intervention. Permanent files cannot be destroyed accidentally during normal system operation, including normal deadstart; they are protected by the system from unauthorized access according to the privacy controls specified when they are created.

Any file attached to a job, regardless of mode or content, which is not already permanent, can be made permanent on a valid rotating mass storage device specified by the installation. Unless the user explicitly requests the system to catalog a file, it will not be made permanent.

Permanent files should be created on devices the installation designates for permanent files by specifying PF device type parameter on a REQUEST card.

TERMS AND CONCEPTS

Terms and concepts used throughout this chapter are defined as follows:

Access Permissions

All user files have a 4-bit permission code. Each bit represents an access permission as defined below:

READ permission: Required to read a file, load a file or copy a file.

MODIFY permission: Required to rewrite or evict part of a file. If the file is organized to contain direct access or indexed sequential files, modify permission is synonymous with use of the replace macro, provided the file was so declared by the FO parameter at catalog time.

EXTEND permission: Required to decrease or increase the amount of mass storage allocated to a particular file and to store position information for a file.

CONTROL permission: Required to purge a file, or catalog a new cycle.

Files in use by a job, other than permanent files, will have all access permissions. Permanent files will have only those permissions granted by ATTACH card parameters. A purged permanent file, when still attached to the job that purged it, has only those permissions it had as an attached permanent file.

Alter

The user can decrease or increase the amount of mass storage allocated to a particular file.

Attach

A permanent file is assigned to a job. No user may access a permanent file until his right to access the file is established and the file is attached to his job.

Archived Permanent File

A file that has been dumped to tape by a permanent file utility routine and is no longer on mass storage, but table information has been retained on mass storage. The system will retrieve the file and re-copy it to mass storage when a user issues an attach request.

Catalog

A file's location and other pertinent information is entered in the permanent file directory maintained by the system, thereby making the file permanent. The first cycle of a file cataloged is called an initial catalog. Any subsequent catalogs under the same permanent file name and owner ID are new-cycle catalogs.

Cycle

Up to five files may be cataloged under one permanent file name. Each is called a cycle. In normal usage, a cycle is one version of a permanent file. Each file shares the same user ID and set of passwords. No restrictions are imposed on the content or size of any cycle, as each is a unique file.

Each cycle is identified by the combination of permanent file name, cycle number, and owner ID. Cycle numbers from 1 to 999 (decimal) can be assigned by the user or the system.

A cycle number assigned by the system will be one larger than the current largest cycle number, but it may not exceed 999. The system automatically assigns a cycle number in the following cases:

Invalid cycle number (0 or greater than 999)

Cycle number not specified (no CY parameter)

Duplicate of existing cycle number

If a user permits the system to assign a new cycle number in new-cycle catalogs, he must recycle the number with the RENAME function after it reaches the value of 999.

Detach

A permanent file is detached at job termination, unless it is detached earlier through the RETURN or UNLOAD control cards or the CLOSE system macro. The CLOSE system macro must specify UNLOAD or RETURN.

Exclusive Access

When only one job may manipulate or read a permanent file, it has exclusive access. Control permission assures exclusive access. Modify or extend permission may grant exclusive access depending on the setting of the RW parameter.

Extend

This operation allows a job to increase the amount of mass storage occupied by an attached permanent file during job execution after writing at end of information.

Highest Cycle

The permanent file cycle with the highest numeric value. The lowest cycle has the lowest cycle number.

Incomplete Cycle

A cycle for which permanent file table information is incomplete. This condition can result from a CATALOG job that does not terminate normally.

Logical File Name (lfn)

The name used by a job to reference an attached permanent file. It is 1-7 alphabetic and/or numeric characters, beginning with a letter; it may be the same as the permanent file name.

Modify (Rewrite)

This operation changes the data content of the file but not the location and length of the file.

Multi-Access

Three types of multi-access are available:

> Multi-read access: any number of jobs may attach a permanent file simultaneously for read only access.
>
> Multi-modify access: any number of jobs may attach a permanent file simultaneously for read access or modify access. Availability of this feature is determined by the installation. If it is allowed, the users must be aware of where the files are rewritten and read. CAUTION IS ADVISED.
>
> Multi-read access with a single extend or single modify access: any number of jobs may attach a permanent file simultaneously for read access while one job attaches for extend or modify access.

Owner

The user responsible for establishing a new permanent file name in the system is identified by the owner ID which he supplies.

Passwords

The owner of a permanent file can define permissions to be granted at attach time.

Passwords can be any string of 1-9 numbers or letters. Each password implies one type of access permission. If a password is not defined for read, modify, extend or control, access for that permission is given automatically to the requestor.

In addition, a fifth password, turnkey, may be defined. This password provides an extra measure of control over file access. When the turnkey password is defined, no permission is granted unless the turnkey password is submitted.

Passwords are not submitted for verification in the same manner as they are defined. The password definition parameters are XR, TK, RD, MD, EX, and CN. The keyword parameter for submitting passwords for verification is PW. These two sets of keywords are exclusive.

Permanent File

A file on a rotating mass storage device defined in the system as a permanent file device. It is standard in mode, content, and length, and locatable by a system search of the permanent file tables. Each permanent file is identified by a permanent file name, owner ID, and cycle number.

Permanent File Directory (PFD)

Table containing a record of all permanent files, their cycles and passwords.

Permanent File Catalog
Record Block Table Catalog (RBTC)

Table containing a record of the physical location of a file and of all statistics associated with that file.

Permanent File Manager (PFM)

System routines that handle user oriented permanent file functions. These functions are available to the user through control cards, system macros and RA+1 calls.

Permanent File Name

The name by which a file is known to the permanent file manager, 1-40 alphabetic and/or numeric characters.

Permanent File Privacy, Security, and Protection

Privacy in permanent files is intended to minimize software interference by thwarting threats to the data base (permanent files) from non-authorized central processor programs. The permanent file system offers a standard set of privacy controls. If an installation requires a different kind of protection, a privacy procedure may be defined to replace the standard.

The system automatically ensures that no normal operation will cause permanent mass storage files to be overwritten or otherwise destroyed, and that the directory of permanent files will not be destroyed.

In addition to normal system protection, the individual file owner can prevent unauthorized access to his permanent file. He can stipulate, in cataloging a file, the degree to which the file is to be protected from read, write, and rewrite access. Once a file is cataloged, it cannot be used by any job unless the necessary passwords are given when a request is made to attach the file.

Permanent File Queueing

If a job cannot attach a file immediately, it will attempt to enter that file in a queue. Four conditions can cause a job making a permanent file request to be placed into the permanent file queue:

A permanent file utility is running

A permanent file table necessary for CATALOG or ATTACH is full

File to be attached is not available for type of access requested

File to be attached is archived

Public File Password

SCOPE provides a password which, if submitted at catalog time, allows a user to catalog a file with the ID of PUBLIC and then omit the ID parameter on new-cycle catalogs, attaches, and purges.

Purge

Removes a permanent file's control information from the directory. A purged file is no longer a permanent file, but it remains available for use by the current job.

Retention Period

When a permanent file is cataloged, a retention period may be specified. The file is not purged automatically upon expiration; however, an installation can obtain a listing of all expired files not purged from the system.

Universal Permission
Universal Password

Installations may define a universal password that gives universal permission to access all permanent files when the password is submitted. The specific permissions granted depends on installation selections.

Write

When a permanent file is attached to a job and positioned at end of information, any write function, except a rewrite, will add data to the file and extend end of information. Such an extension is considered temporary; unless the EXTEND or ALTER function is performed successfully, the extension is erased when the job is terminated.

PERMANENT FILE CONTROL CARDS

All permanent file control cards are written in one of the following forms:

function name(lfn,pfn,parameter list)

function name(lfn,parameter list)

function name(pfn,parameter list)

The seven function names are:

CATALOG, ATTACH, EXTEND, RENAME, PURGE, SETP, ALTER

lfn is the logical file name; 1-7 characters; first character must be alphabetic; position dependent.

pfn is the permanent file name; 1-40 letters or numbers; position dependent.

All parameters in the list are written in the form:

keyword = value

Keyword is always 2 characters; value may be a decimal number or a string of 1-9 alphabetic and/or numeric characters.

Permanent file control statements may be continued on as many cards as needed to contain the function parameters. If a parameter list has no terminator (right parenthesis or period), column 1 of the next card is considered as the immediate continuation of column 80.

PARAMETERS FOR CONTROL CARDS AND MACROS

The character strings (keyword = value) described below may be written in any order on control cards.

File identification parameters:

ID = name   Name, 1-9 numbers or letters which identify the file creator (or owner). Permanent file names file names are unique to owner IDs. Default is an ID of PUBLIC. An ID of SYSTEM is reserved for system use.

CY = n   Cycle number, 1-999, assigned by file creator. Default value on a new-cycle CATALOG is one higher than the current highest cycle number, not exceeding 999. On an attach, it is the highest cycle number cataloged. n = 0 may be specified, but it is ignored.

LC = n   If n is non-zero, the lowest cycle is referenced. Default is the highest cycle number.

Installation parameters:

RP=n Retention period in days, 0-999, as specified by file creator; infinite retention is indicated by 999. Installation may define default value to be less than infinite.

AC=name 1-9 alphabetic and/or numeric characters to specify an account parameter.

PP=name Information to be passed to an installation defined privacy procedure, 1-9 characters.

Password definition parameters:

Passwords are formed from 1-9 alphabetic and/or numeric characters. A zero length character string is accepted on the RENAME control card to remove a password definition.

TK=pw Turnkey password

RD=pw Read password

EX=pw Extend password

MD=pw Modify password

CN=pw Control password

XR=pw Password definition for all passwords except read, turnkey, and any specifically defined.

Permission generation parameters:

PW=list Maximum of five passwords on ATTACH control card establish user's access permission for a file.

PW=pw1,pw2,...pw5

PW is required also on the CATALOG card when a new cycle is added, on the PURGE card under permanent file name mode. and when a file is cataloged with an ID of PUBLIC. PW is the only parameter that may be used to submit passwords at attach time.

MR=n If n is non-zero, the file will be granted read permission only (if read permission password has been given), thereby encouraging multi-read access.

RW=n If n is non-zero, multi-read with single rewrite or single extend will be allowed. If the installation allows multi-modify access, multi-read with multi-rewrite will be allowed. Control permission will not be granted to a job specifying RW not equal to zero. If n is zero, a job will be given exclusive access if it has control, modify, or extend permission.

File structure parameters (applicable to Record Manager files only):

FO=DA
FO=IS — If DA or IS is specified, a data validity check is made for indexed sequential or direct access files. Extend and modify permissions are given logical meaning for such files by use of this function.

Positioning parameter:

PS=n — If non-zero, a file is attached at a position previously set by the SETP function. If absent, equal to 0, or invalid because of an ALTER function, the default is used and the file is attached rewound. If used in a purge by permanent file name request, the file will remain positioned after a successful purge.

ECS buffering parameters:

EC=K — Installation standard number of blocks of ECS for buffer.

EC=nnn
EC=nnnnK — Number of thousand-word ECS blocks (octal) to be allocated.

EC=nnnnP — Number of ECS pages (octal) to be allocated.

The following have been included for compatibility with previous systems:

SD=n — ignored

RN=n — ignored

PERMANENT FILE FUNCTIONS

The permanent file functions are written as SCOPE control cards. They also are available for use as COMPASS program macro calls. All parameters may be given on control cards; however, only relevant parameters will be recognized. Others not pertinent to the function will be ignored. Functions and parameters are summarized in the following table.

	lfn/pfn	AC	CN	CY	EC	EX	FO	ID	LC	MD	MR	PP	PS	PW	RD	RP	RW	TK	XR
CATALOG	one or both	*	*	*		*	*	+		*	*	*		*	*	*	*	*	*
ATTACH	one or both			*	*			+	*		*	*	*	*			*		
PURGE	one or both			*	*			†	*			*	*	*					
RENAME	lfn	*	*	*		*		*		*				*	*	*		*	*
EXTEND	lfn																		
SETP	lfn																		
ALTER	lfn																		

+ required * optional † special case

CATALOG FUNCTION

The CATALOG function makes permanent a file which has all permissions granted to it. A new cycle catalog requires only control permission to be established.

```
CATALOG(lfn,pfn,parameter list)
```

The following optional parameters are relevant on a CATALOG function:

Initial catalog:	CY,XR,CN,MD,EX,RD,TK,RP,FO,RW,MR,PW,AC,PP
New-cycle catalog:	CY,PW,RP,FO,RW,MR,PP,AC

For this function,

User must:
- Identify a local file to be made permanent
- Specify his owner ID
- Define a permanent file name unique to his ID
- Be granted control permission to add a new cycle

User may:
- Define a cycle number
- Define passwords to control access permission generation
- Define the retention period
- Define the permanent file as an indexed sequential or direct access file.
- Remove some permissions

CATALOG Examples:

The first set of examples demonstrate initial catalogs; the permanent file name is unique to the ID specified.

1. `CATALOG(LFN,LFN,ID=RENOIR)`

   `CATALOG(LFN,ID=RENOIR)`

   These statements achieve the same effect. Any time the permanent file name is omitted, it will be assumed to be the same as the logical file name. The cycle number will be one.

2. `CATALOG(LFN1,PERMANENTFILE,ID=RENOIR,CY=10)`

   The first cycle cataloged may have a cycle number greater than one.

3. `CATALOG(LFN2,PFILE,ID=RENOIR,CY=0)`

   The cycle number of the permanent file, PFILE, will be one since an illegal cycle number is specified. The cycle number must be 1 through 999. Otherwise, the parameter is ignored.

4.
```
CATALOG(WATER,LILIES,ID=CMONET,XR=X)
```

```
CATALOG(WATER,LILIES,ID=CMONET,MD=X,CN=X,EX=X)
```

   These two control cards above demonstrate the XR parameter and have the same effect. X will be the password for control, modify and extend access.

5.
```
CATALOG(AA,B,ID=SEURAT,XR=Y,CN=Z)
```

```
CATALOG(AA,B,ID=SEURAT,MD=Y,EX=Y,CN=Z)
```

   These two control cards have the same effect, further demonstrating use of the XR parameter.

6.
```
CATALOG(C,F,ID=SIGNAC,FO=IS,MD=X,EX=Y)
```

   If a data validity check reveals the file is an indexed sequential or direct access file, extend permission will become insert permission, and modify permission will become replace permission. If the file is not an IS or DA file, the FO parameter is ignored.

7.
```
CATALOG(LFF,PF,ID=MATISSE,RP=5,CY=4,RD=X,CN=Y,MD=A, TK=C,AC=777,PP=XYZ,MR=1)
```

   Since the MR parameter is non-zero, LFF will have only read permission upon catalog completion. The following items are defined at catalog time:

Read password	X
Control password	Y
Modify password	A
Turnkey password	C
Account parameter	777
Cycle number	4
Retention period	5 days
Privacy procedure	XYZ

Assuming the previous examples to be successful initial catalogs, the following examples demonstrate new-cycle catalogs. A file already has been cataloged with the permanent file name and ID specified.

8.
```
CATALOG(Z,LFN,ID=RENOIR)
```

```
CATALOG(Z,LFN,ID=RENOIR,CY=2)
```

   These control cards catalog a cycle with a cycle number one higher than the largest (in this case 1). This new-cycle catalog does not require passwords because a control password was not defined.

9.
```
CATALOG(LFN22,PERMANENTFILE,ID=RENOIR,CY=10)
```

   Assuming, a cycle 10 already exists, this control card will cause a cycle 11 to be cataloged. An invalid cycle number is treated as no cycle number. This new-cycle catalog does not require passwords because a control password was not defined at initial catalog time.

10.
```
CATALOG(LFF,PF,ID=MATISSE,CY=5,PW=Y)
```

   If a control pasword is defined at initial catalog, it is necessary to submit the control password using the PW parameter. Control permission is required to add a new cycle.

11. `CATALOG(LFF,PF1,ID=PUBLIC,PW=XYZ)`

    A file may be cataloged with an ID of PUBLIC if the public password is submitted–defined by the installation as XYZ in this example. This enables an installation to define permanent files that may be attached by all users without specifying an ID.

12. `CATALOG(PERMANENTFILENAME,ID=MOREAU)`

    A catalog function will be attempted using the first seven characters of the permanent file name as the logical file name. If the logical file name is omitted, the first character of the permanent file name must be alphabetic, or the job will be terminated.

ATTACH FUNCTION

The ATTACH function requests attachment of a permanent file to a job and establishes the requestor's legal access to the file. Evaluation of the cataloged passwords and the password list submitted with the request establishes the type of access granted to the user. When access is ascertained, the permanent file is attached to the job and it may be used only as specified by the ATTACH function. For example, a file attached with only read permission may not be modified or extended.

```
ATTACH(lfn,pfn,parameter list)
```

In catalog example 7, a permanent file, PF, was cataloged with turnkey, modify, control and read passwords. The following examples demonstrate permission generation using the PW parameter:

`No PW parameter`	No permissions
`PW=Y,A,X` or `PW=Y`	No permissions
`PW=X` or `PW=A,X` or `PW=A`	No permissions
`PW=C`	Extend permission
`PW=C,A`	Modify and extend permission
`PW=X,Y,C,A,MR=1`	Read permission
`RD=X,PW=Y,C`	Extend and control permissions
`PW=X,Y,C,A,RW=1,MR=1`	Modify, read, extend

Unless a file has an ID of PUBLIC, the ID parameter always must be specified to uniquely identify a file. Only the installation can catalog files with the ID of PUBLIC. Only the system can catalog files with the ID of SYSTEM.

The following optional parameters are relevant on an ATTACH function:

LC or CY, PW, PS, MR, RW, PP, EC

For the ATTACH function,

User must: Specify the owner ID of the file unless the file has ID of PUBLIC.

Identify the permanent file and specify the LC parameter if the lowest cycle is required and more than one cycle exists.

User may: Specify a cycle number, or specify the lowest cycle required.

Submit passwords using the PW keyword; attach the file positioned.

LC and CY are conflicting keywords; if both are specified, LC is ignored.

When an incomplete cycle is to be attached, the CY parameter must be specified and control permission must be established. After the ATTACH is complete, the file may be purged.

ATTACH Examples:

1. `ATTACH(LFN,ID=RENOIR)`

```
ATTACH(LFN,LFN,ID=RENOIR)
```

Assuming, catalog example 8 was successful, these two control cards perform the same function. If the permanent file name is omitted, it is assumed to be the same as the logical file name. Cycle 2 will be attached since that is the highest cycle number.

2. `ATTACH(LFA,PF,ID=MATISSE,PW=X,C ,EC=K)`

Assuming catalog example 7 was successful, cycle 4 of the permanent file, PF, will be attached with read and extend permission. During execution the permanent file is referred to by the logical file name, LFA. A standard size ECS buffer will be establshed for the file.

3. `ATTACH(LFILE,PF,ID=RENOIR,PS=1)`

All permanent files are attached rewound if the PS parameter is not specified. If the PS parameter is specified, the permanent file tables are checked to determine if an attach position was set for this file by a SETP command. If not set, or the position has been invalidated by the ALTER function, the file is attached rewound.

4. `ATTACH(PERMANENTFILENAME,ID=RENOIR)`

An attempt will be made to attach the permanent file, PERMANENTFILENAME, under the logical file name, PERMANE. The first seven characters must be letters or numbers and begin with a letter if the local file name is omitted in the attach call.

RENAME FUNCTION

Changes can be made to the permanent file name, cycle number, passwords, retention period, account parameter, and owner ID. However, the permanent file name, ID, and CY cannot be changed if any of the file cycles have been dumped or placed on an archive tape. The RENAME function cannot be performed unless the file has been attached and all four permissions (READ, EXTEND, MODIFY, and CONTROL) have been granted. The control card is written.

```
RENAME(lfn,pfn,parameter list)
```

lfn must specify an attached permanent file with all permissions. Remaining parameters are optional. If none are specified, the control card is processed as a no-operation. If the permanent file name is not to be changed, one or two commas should follow the lfn (2 commas are allowed for compatibility with previous systems).

RENAME Examples:

1. Assume PFILE was cataloged by owner ABC with read password X, extend password Y, and modify password Z. Control is granted automatically.

   ```
   ATTACH(LFILE,PFILE,ID=ABC,PW=Y,Z,X)
   RENAME(LFILE,PFILE2,RD=,CN=W)
   ```

   The permanent file name PFILE will be replaced by PFILE2. The read password will be removed (succeeding users will be given read permission automatically) and a password for control permission will be cataloged. The existing passwords for extend and modify will remain unchanged. Since the changes involve the permanent file name and passwords, the changes apply to all cataloged cycles of the file. This would also have been true if the owner ID had been changed.

2.
   ```
   ATTACH(LFN,ID=UTRILLO)
   RENAME(LFN,,ID=UTRILLO,RD=A,RP=9)
   RENAME(LFN,LFN,ID=UTRILLO,RD=A,RP=9)
   ```

   This RENAME control card defines a READ password for the permanent file LFN, and redefines the retention period. Omission of the permanent file name in the RENAME control card indicates no name change is to occur. The two RENAME control cards are identical in function. This example also demonstrates that more than one RENAME function can be issued consecutively.

3.
   ```
   ATTACH(LFN,,ID=SISLEY,PW=A)
   RENAME(LFN,,ID=SISLEY,RD=)
   ```

   The definition of A as the READ password is removed from the permanent file, LFN.

EXTEND FUNCTION

Permanent extensions can be made to a permanent file by writing at end of information and cataloging extensions to the file. SCOPE requires extend permission for this operation.

To catalog extensions, the EXTEND function must be used on that permanent file. Also, a file cataloged by a given job may be extended by that job. The control card is written:

```
EXTEND(lfn)
```

The lfn parameter is required; it is the logical file name of the attached file to be extended. The newly added section will acquire the privacy controls of the permanent file.

If lfn refers to an indexed file, the current index must be rewritten, at the end of the file, by the user to invalidate any prior index. This must be done prior to the EXTEND function. When an EXTEND function is requested for a random file, nothing must have been written on the file since the index was last written.

No keywords are relevant to EXTEND.

PURGE FUNCTION

A cycle of a file can be removed from the catalog of permanent files. The control card is written:

```
PURGE(lfn,pfn,parameter list)
```

or

```
PURGE(lfn)
```

If the logical file name on the PURGE card references an attached permanent file, only the lfn parameter is used. Otherwise, control permission must be established before a request for a PURGE function is granted. All parameters applicable to attach are then applicable to PURGE.

PURGE Examples:

1. `ATTACH(LFN,ID=RODIN)`

   `PURGE(LFN)    or    PURGE(LFN,ID=RODIN)`

   Both sequences perform the same function.

   When a PURGE is performed, permanent file table information for the file will be removed, but the file will remain available to the job with permissions existing when it was purged. At least control permission is implied.

2. `PURGE(PERMANENTFILENAME,ID=PISSARO)`

   The permanent file, PERMANENTFILENAME, will remain attached to the job as a non-permanent file if the PURGE is successful. The logical file name,PERMANE, will reference that file. The PURGE will not be successful if the logical file name is omitted in the call and the first character of the permanent file name is not alphabetic.

3. `PURGE(PERMANENTFILE,ID=RENOIR,LC=1)`

   Assuming catalog examples 2 and 9 were successful, cycle 10 will be purged and thereafter will be known to the job as the non-permanent file, PERMANE.

ALTER FUNCTION

The ALTER function causes end of information to be set to current PRU.

```
ALTER(lfn)
```

lfn specifies the logical file name of an attached permanent file.

The ALTER function always requires extend permission; modify permission and exclusive access to the file are required if the current PRU precedes end of information of the file as it was attached. The user can ensure exclusive access by specifying RW = 0 on the ATTACH request.

The ALTER function is identical to the EXTEND function if an extension has been made to the attached file and the current PRU is at the new end of information.

SETP FUNCTION

SETP is a positioning function that stores a current PRU for an attached permanent file in the PF tables. Subsequent attaches can specify a nonzero PS parameter value so the file will be positioned to the PRU specified by the previous SETP. If a SETP function was never done, the position has been invalidated by an ALTER function, or the PS parameter value is zero, the file is attached rewound.

```
SETP(lfn)
```

The SETP function always requires extend permission.

SETP Example

```
ATTACH(A,B,ID=SEURAT,PW=Y)
SKIPF(A,1)
SETP(A)
RETURN(A)
ATTACH(A,B,ID=SEURAT,PS=1)
```

Assuming initial catalog example 5 took place, the first statement will attach the permanent file B with all permissions. The file will be attached rewound. The attached file is positioned by using SKIPF. A SETP function then is issued, causing the PRU position to be recorded. The next ATTACH after the RETURN will cause the file to be attached after the first PRU because the PS parameter is specified. If PS had not been specified, B would have been attached rewound, regardless of whether a SETP function had been issued on the permanent file B. The SETP function recorded a PRU position only for the attached cycle. Other cycles of permanent file B remain unaffected.

PERMANENT FILE MACROS

Each permanent file macro expansion contains a call to a permanent file PP program. Parameters necessary for execution of a function are contained in the central memory table called the file definition block (FDB).

The macro for generating an FDB has the format:

```
   fdbaddr     FDB   lfn,pfn,parameter list
```

fdbaddr is the symbol to be associated with word 5 of the FDB, and it must be present in the location field. Parameters are separated by commas and terminated by a blank. They may include any of those indicated by the two-letter codes described for control cards.

The field to the right of the macro name, FDB, is identical to that which could be on a control card. Parameters are entered into the FDB as they are encountered in the list. The FDB is generated in-line during assembly whenever the macro is called.

The macro function call is of the following form:

```
   function fdbaddr,RC,RT,NR
```

fdbaddr is the symbol on the FDB macro; function is any permanent file function, such as CATALOG.

If the RC or RT parameter is specified in the function call referencing an FDB, a return code will be available to the user in word 5 of the FDB, - at fdbaddr (bits 9-17). In addition, if specified, the RT parameter will inhibit any permanent file queueing. All permanent file macro calls will be issued in recall unless NR is present. In this case, it will be possible for the central processor program to test the completion bit in the FDB to determine whether the function has completed.

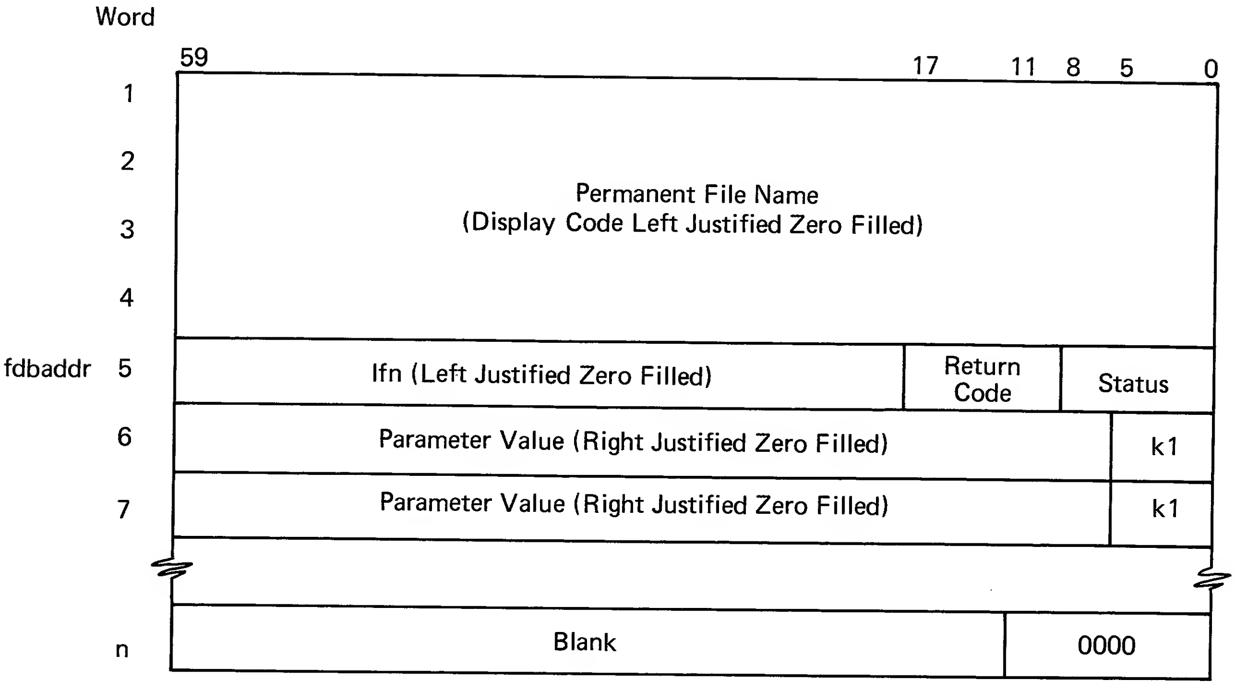
The RT parameter can be used only at the macro level, unless the user constructs the FDB himself. Jobs attempting permanent file attaches will queue for the requested file if the RT parameter is not specified when:

File is unavailable (job requesting file wants exclusive access, or job using the file has exclusive access).

Attached permanent file table is full.

Archived file (a permanent file stored on tape rather than mass storage) is temporarily unavailable. ATTACH will set up a LOADPF job to be scheduled through tape scheduling. The job requesting the attach will be swapped out until the file is available.

The FDB generated by the macro has the form:


Word
59
17
11
8
5
0
1
2
3
4
Permanent File Name
(Display Code Left Justified Zero Filled)
fdbaddr
5
lfn (Left Justified Zero Filled)
Return Code
Status
6
Parameter Value (Right Justified Zero Filled)
k1
7
Parameter Value (Right Justified Zero Filled)
k1
n
Blank
0000

k1 2–digit octal number identifying parameter in list shown below

Status Bit 0 Complete bit
1 Not used
2-5 Function code listed below
6 Set if RC or RT not specified – issue dayfile messages – all errors fatal
7 RT specified
8 Issue dayfile messages

Return Code 9 bit code listed below

4-bit function codes (bits 2-5) of status field:

ATTACH	0010	RENAME	1010
CATALOG	0100	PERM	1100
EXTEND	0110	SETP	0001
PURGE	1000	ALTER	0111

Values for k1 correspond to the parameters in the FDB macro. Representation appears in binary unless otherwise noted.

k1 Value Octal	Parameter	Description of Macro Field
01	PP	Privacy procedure name (display code)
02	RP	Retention period days
03	CY	Cycle number
04	TK	Turnkey password (display code)
05	CN	Control password (display code)
06	MD	Modify password (display code)
07	EX	Extend password (display code)
10	RD	Read password (display code)
11	MR	Multi-read parameter
12	SD	Ignored
13	XR	Control, Modify, Extend password (display code)
14	ID	Owner identification (display code)
15	RN	Ignored
16	AC	Account
17	EC	ECS Buffering
20 24	PW	Password submitted (display code)
25	FO	File organization (display code)
26	PS	Position
27		Ignored
30		Ignored
31	LC	Lowest cycle
32		Ignored
33	RW	Multi-access rewrite
34-37		Ignored (reserved for SCOPE)
40-47		Ignored (reserved for user/installation)
50-77		Ignored (reserved for SCOPE)

The 9-bit return code at fdbaddr (bits 9-17) can take the following values:

User Return Code	Description
00	Function successful
01	ID error
02	lfn already in use
03	Unknown lfn
04	No room for extra cycle
05	RBTC full
06	No lfn or pfn
10	Latest index not written
11	File not on PF device
12	File not in system
15	Cycle number limit reached
16	PFD full
17	Function attempted on non-permanent file
20	Function attempted on file not part of job
22	File never assigned to a device
23	Cycle incomplete or dumped
24	PF already attached
25	File unavailable
27	Illegal lfn
33	ALTER needs exclusive access
35	File already in system
70	PFM stopped by system
71	Incorrect permission

On control card requests, all errors are fatal; and on macro requests unless the RC or RT parameter is specified, the job is terminated. An installation parameter determines whether errors which return codes greater than octal 67 will cause termination on a macro request. All internal permanent file malfunctions are system errors which will cause job termination.

PERM FUNCTION

PERM, available only as a system macro, allows a running program to determine what permissions have been granted to a file and whether or not the file is permanent. The macro format is:

```
PERM   fdbaddr,RC
```

The only required parameter is the lfn which should be supplied in the FDB. This lfn should reference a file available to the job calling PERM.

The 5-bit code is returned to the user as the return code in the FDB (bits 9-13 in fdbaddr). The rightmost 4 bits are the permission bits. The octal codes are:

10 CONTROL
04 MODIFY
02 EXTEND
01 READ

The leftmost bit of the 5-bit code is a flag. If it is 0, the lfn references an attached permanent file unless the entire 5-bit code is equal to 0. If the code is 0, the lfn is unknown to the job calling PERM since a permanent file cannot be attached without permissions.

PERMANENT FILE USAGE VIA INTERCOM

From his terminal, the INTERCOM user may create, attach, and purge permanent files in any of three ways:

Through the standard macros within his own interactively run program.

By entering the commands ATTACH, CATALOG, etc. as if they were control cards in a batch input file.

By using the special INTERCOM commands FETCH, STORE and DISCARD. These commands allow the user to create and use permanent files with certain restrictions.

Files created by the STORE command may not have any passwords. The only parameters for STORE are: filename, user id, privacy procedure. The permanent file name and the local file name are the same, user id and/or privacy procedure are required according to installation options. If a required parameter is missing, it will be requested from the user.

When a permanent file has been created through the STORE command, the user may access it through the ATTACH or FETCH commands. FETCH parameter requirements are the same as for STORE.

Similarly, the DISCARD command as well as the PURGE command may be used to purge a permanent file created by the STORE command. DISCARD has the same parameter requirements as STORE, with the exception that the user ID and privacy procedure parameters may be omitted if the file is already attached to the user.

Examples:

In these examples the information output by the INTERCOM system on the teletypewriter terminal is underlined to distinguish it from that entered by the user; this does not actually occur on the teletypewriter output. The symbol ∧ denotes carriage return.

The installation requires both user id and a privacy procedure parameter; the user file called MYFILE is to be made permanent.

```
COMMAND-STORE,MYFILE.∧

ID=RKC∧

PP=167SN58∧
```

During a later session the user attaches the file and then purges it.

```
COMMAND-FETCH,MYFILE,RKC,167SN58.∧

COMMAND-DISCARD,MYFILE.∧
```

If an INTERCOM job enters into the permanent file queue because a permanent file request cannot be honored immediately, the user will be informed by one of the following messages:

WAITING FOR PF UTILITY

WAITING FOR APF SPACE

WAITING FOR ACCESS TO FILE

WAITING FOR ARCHIVED FILE

If an attach is requested for an archived file, the operator will be asked for a GO or DROP response to determine if file retrieval should be initiated. The INTERCOM user will be informed of the delay by the message:

REQUEST FOR ARCHIVED FILE - WAITING FOR CENTRAL OPERATOR DROP OR GO

In response to GO, the job will be put into the permanent file queue, the message WAITING FOR ARCHIVED FILE will be sent to the terminal user, and a job will be set up at another control point to retrieve the file from tape. The INTERCOM user must wait for retrieval to be completed before the file is attached. In response to DROP, the file will not be brought into the system and the attach request wi!l be terminated.

If the user does not want his job to remain in the permanent file queue until the requested file can be accessed, he may terminate the attach request by typing %A at a CRT terminal, or at a Teletype, CTRL Z followed by A.

USER COMMUNICATION

A user specifies the intent of a particular directive by specifying parameters. If they do not clearly define the function request, the permanent file manager attempts to inform the user of the unknown information by the following means:

> Modification of the File Definition Block will be done when an illegal parameter is correctable. For example, if an incorrect cycle number is encountered on a CATALOG card, the actual cycle number will be returned in the FDB. If code is not successful, error codes may be returned in the FDB.
>
> A job dayfile message will be issued to the job dayfile unless the RT or RC parameter is specified. (Appendix D.)

If a permanent file control card is processed without an error diagnostic, the function was successful and all parameters were used as specified on the control card. If a macro call did not specify RC or RT and it finished without an error diagnostic, it was successful. A macro call issued with RC or RT, was successful if a zero code is returned.

A message will be issued to the INTERCOM terminal if queueing has been initiated because a file is unavailable or the APF table is full. The user may then respond whether or not he will wait.

LOADER AND LIBRARIES 6

SCOPE 3.4 provides extended features in library organization including:

Multiple system libraries

Multiple user libraries

Selection of library search sequence

In addition, the SCOPE 3.4 system offers a NUCLEUS library which may be modified according to installation needs.

Segments have been completely redefined. Multiple level overlays are now possible.

The loader is now one entity. PPLOADR and CPLOADR no longer exist. A full description of the loader with all control cards and macros appears in the LOADER Reference Manual .

An overview of the new loader and its relationship to the new library structure follows. This overview, although not complete, is included to acquaint the user with the principles and terminology used in the new loader.

LOADING PROCESS

Loading places object code into core making it ready for execution. The object code is fetched from local files and libraries. Normally, object code is output from a compiler or assembler; but it may come from another processor, such as an overlay generator. When loading is completed, program execution may be initiated.

Loading also involves related services, such as presetting unused core and writing a map to show what has been loaded.

SINGLE-MODULE LOADING

When all object code is loaded into core at the same time, resulting in a single core image module, loading is single-module (also referred to as normal mode, or non-segmented, loading.) Most programs are loaded in this manner, as they are relatively small compared to the amount of available core.

The basic loader performs single-module loading.

MULTIPLE-MODULE LOADING

Some programs are too large to fit in the largest available field length. Also, single-module loading of large programs may waste core because space is taken by program portions not always in use. Multiple-module loading is provided, so that only certain portions of the executing program need be in core at any one time; the total program consists of more than one core image module. At different times, the same area of core may be occupied by different modules.

The two types of multiple-module loading are overlays and segments.

OVERLAYS

The overlay structure is compatible with loaders of earlier SCOPE versions. Three levels are possible; 63 overlays are allowed at the second level, and 63 are allowed at the third level for each one at the second level. Loading of overlays is not automatic;,explicit calls must be made to load an appropriate overlay.

SEGMENTS

The definitions of segment and segment loading have been changed for SCOPE 3.4. Segments now are rather like overlays as described above; but a very large number of levels is allowed; external symbols and common block names can refer to higher as well as to lower levels; and segments are loaded automatically when an entry point of a segment not in core is referenced by an instruction in another segment.

RELOCATABLE LOADING

Relocatable loading connects contiguous storage areas (blocks). Blocks are defined by relocatable object modules which contain relocation information so that their origins need not be at a particular location in core. During relocatable loading, the origins of all blocks are established and all addresses are adjusted accordingly. It is possible to load relocatable object modules produced by one or more independent compilations or assemblies. The following terms relate to relocatable loading:

RELOCATABLE SUBPROGRAM

The basic program unit produced by an assembler or compiler. In COMPASS, it is produced as a result of the source statements delineated by IDENT and END. In FORTRAN, the source statements are specified between PROGRAM, SUBROUTINE, BLOCK DATA, or FUNCTION and END. A relocatable subprogram consists of several tables which define blocks and information needed to relocate addresses.

COMMON BLOCK

Common blocks differ from program blocks in that the same block may be declared by more than one relocatable subprogram. More than one relocatable subprogram may specify data for the same common block; but in case of an address conflict, information will be loaded over previously loaded information. A program may declare as many as 509 common blocks. Common blocks may be labeled or blank.

Labeled common is a common block into which data may be stored at load time. Depending on the type of source statement, a labeled common block may specify either CM or ECS for storage. The first program declaring a particular labeled common block determines the amount of central memory or ECS allocated. Should a subsequent declaration of the same block indicate a larger size, a non-fatal diagnostic will be issued; and the original declaration will hold. No additional checks are made to determine whether larger subsequent declarations result in text loading that overflows into higher blocks. A CM or ECS block can have the same name.

Blank common is common block storage into which data cannot be stored at load time. The first declaration of a blank common block need not be the largest. In single-module loading and segment loading, one central memory blank common and one ECS blank common block are allowed. After all object modules have been processed, the central memory block is allocated storage at a higher address than all other blocks. In overlay loading, any one overlay can have one central memory blank common block, and one ECS blank common block; and more than one overlay can have such a block; but an overlay cannot have a central memory blank common block if its parent overlay, or the (0,0) overlay, has one, and similarly for an ECS blank common block.

ABSOLUTE INFORMATION

For basic single-module loading, a relocatable subprogram may contain information to be stored at a specific location in central memory. Generally such information should only be stored in the communication area below octal location RA + 100. Results cannot be predicted if the origin of absolute information is higher than this value.

ENTRY POINT

A location within a block which may be referenced from other blocks. Each entry point is associated with a name. The loader keeps track of entry points through the entry point tables in relocatable subprograms. A block may contain any number of entry points.

The loader can accept entry point names of 1-7 characters; none of which may be a colon.

EXTERNAL REFERENCE

An address expression which consists of a reference to an entry point in another block. Throughout the loading process, externals are linked to entry points. In some cases, this process is inhibited.

UNSATISFIED EXTERNAL

An external reference that appears before the program containing the matching entry point has been loaded.

TRANSFER SYMBOL

The name of the entry point which specifies where execution is to begin.

PROGRAM NAME

The name contained in the PREFIX table at the beginning of each relocatable subprogram or absolute overlay; the terms ident name or deck name have the same meaning. The loader accepts program names in the same format as entry point names.

ABSOLUTE LOADING

The loading of only one block that is an absolute program or a core image module. The origin of this block must be a particular location in central memory or ECS. Absolute loading involves no block placement, address relocation, or external linking.

An absolute program may be entirely self sufficient, or it may be only one of several core image modules, but it must consist entirely of absolute program. They may be produced directly by a compiler, an assembler, or the loader. The loader produces absolute programs by performing relocatable loads and then writing the core image of the load onto a file.

LOADER INPUT

Two types of information are processed by the loader:

Binary tables: information necessary to perform a relocatable or an absolute load.

Loader object directives: described below.

LOADER REQUESTS

Requests are made to the loader for specific action, such as loading from particular files or selecting options. Most loader requests originate on control cards in the job control stream (first SCOPE logical record of the file INPUT). These requests are processed as they are encountered.

Loader object directives refer to those loader requests encountered as part of the loader input. They are processed as they are encountered during physical loading.

User calls consist of one or more loader requests issued from an executing program.

LOAD OPERATION

The terms load sequence and load operation are synonomous. The following basic steps are involved:

Initialize load

Perform loader requests

Complete load

Load operations initiated by control cards take place as described above. User call load operations initialize only a particular area in core, so that the executing program is not destroyed; user calls can control whether or not the satisfying of externals is to be performed during completion.

The following definitions related to load operation are essential for understanding the loader requests.

INCOMPLETE LOAD

State of load operation from the time loading is initiated until the time execution can begin. Throughout incomplete load, loader information in central memory must not be destroyed if the load is to continue.

LOAD COMPLETION

Steps taken by the loader after all requests have been performed. Normally, the last step is to start program execution. Certain loader requests are defined as the last to be processed before the load is completed. All others are processed in order of occurrence.

LOADED PROGRAM

The core image module; the final image produced by the load operation. For a control card initiated load operation, this module is the entire job field length from RA+100B through RA+FL−1. For a user call initiated load operation, it is only that portion of the field length specified as available in the user call.

LIBRARY SET SEARCH

The order in which libraries are selected for searching is determined by the current contents of a list of libraries known as the library set. Each library so defined is searched in turn until all needs are met or all libraries have been searched. The total library set consists of the global library set and the local library set followed by the library NUCLEUS. The NUCLEUS library cannot be removed from the library set.

Initially, the global library set is empty; it can be altered by the LIBRARY control card. Its definition remains in effect throughout the job or until another LIBRARY control card is encountered.

The local library set is defined only for the duration of the current load operation. At the start of each load operation, it is defined as empty. It can be altered by the LDSET(LIB=...) request.

LIBRARIES

Library structure is comparable to file structure. The loader can access the library randomly by consulting a directory. The user can access system libraries as well as his own libraries and local files.

Libraries are used by the loader to:

> Select programs for execution: absolute programs (overlays) and relocatable programs (object modules)
>
> Select programs to satisfy external symbols during core image generation; they must be in relocatable format.

Library types are system and user:

> A system library is available automatically to all jobs. It is named in the Library Name Table in central memory resident (CMR), it is contained on a permanent file that can be read by more than one job at a time, and parts of it may be contained in CMR.
>
> A user library is a file formatted as a library, but it is not available to a job until it has been explicitly brought to the job. The job might create the file before using it as a library, or it might be a permanent file that a job would attach explicitly. A permanent file might be such that more than one job could read it at once; but the explicit ATTACH would be needed in every job.

LIBRARY SEARCH

SATISFY STATEMENT

Normally, externals are not satisfied until load completion. With the SATISFY statement, the user can direct the loader to satisfy externals at an intermediate point in a load sequence. If libraries are named in the SATISFY statement, those, and only those libraries are used, in the order given. If no libraries are named, the current library set is used.

Unlike LIBRARY or LDSET(LIB=...), SATISFY does not alter the library set.

LIBRARY STATEMENT

A LIBRARY statement declares libraries to be members of the global library set, until the next LIBRARY statement or the end of the job. Libraries to constitute the set are listed in order as parameters after LIBRARY. Each LIBRARY statement annuls the effect of a previous LIBRARY statement in the same job. At the beginning of a job, the global library set is empty.

LIB DIRECTIVE

If the LIB directive does not name libraries, the local library set becomes empty. If it names libraries, they are added to the local library set, after existing library names. At the beginning of a load sequence, the local library set automatically is empty.

LIBRARY SEARCH ORDER

The current library set is searched: first the current global library set in the order specified on the last LIBRARY statement; then the current local library set, in the order named in LIB directives since the beginning of the current load; then NUCLEUS.

Example:

`LGO1.`	Because the default global library set as defined at the beginning of the job is still in effect and no LIB requests have been made, the library set consists of NUCLEUS only.
`LIBRARY(RUN)`	The global library set now contains RUN.
`LDSET(LIB=USER)`	Introduces a local library set consisting of the library USER.
`LGO2.`	Libraries are searched in the order: RUN, USER, NUCLEUS. The global library set is searched first, then the local set and NUCLEUS.
`LGO3.`	Because this is a separate load operation, the local library set as defined no longer exists; however, the global library set as previously defined remains in effect. Thus, RUN and NUCLEUS are searched.
`LIBRARY.`	Creates an empty global library set.
`LGO4.`	Only NUCLEUS is searched.
`LIBRARY(AX,SCOPE,USER2)`	Defines a global library set consisting of three system libraries.
`LDSET(LIB=ALGOL)`	Defines a local library set consisting of one library.
`LGO5.`	Libraries are searched in the order: AX, SCOPE, USER2, ALGOL, NUCLEUS.

Any compiler call forces an internal LDSET directive, bringing the library containing that compiler's object routines into the local library set.

SCOPE 3.4 PRODUCT SET LIBRARIES

The following table gives the default structure of the product set libraries. Each system library structure may be altered to individual installation needs, subject to the following restrictions:

NUCLEUS contains all programs that can be called by control card.

Entry point names must be unique within a single library.

Program names also must be unique within a single library, but a program can have the same name as an entry point in the same library.

If the name of a compiler object routine library is changed, either the compiler itself must be altered to generate the correct LIB directive or users must be instructed to insert LIBRARY and/or SATISFY statements.

PRODUCT SET ENTITIES

PP programs	
Main programs	Level (0,0) overlays and other programs callable by control card
System texts	System text overlays used by COMPASS
Modules	Relocatable subprograms callable by external symbol references from system or user programs
Object library	Modules used during execution of a program compiled by a higher level language

LIBRARY STRUCTURE

Name	Contents
none	PP programs
NUCLEUS	Main programs Primary overlays System texts
SYSOVL	Higher level overlays
SYSIO	6RM modules CPC modules FORM modules SYMPL object library
FORTRAN	FTN 4.0 object library
RUN2P3	RUN 2.3 object library
COBOL	COBOL 4.0 object library SORT 4.0 modules
SYSMISC	COBOL 3.0 object library SORT 3.0 modules FTN 3.0 object library ALGOL object library SIMULA object library SIMSCRIPT object library BASIC object library

USER LIBRARIES 7

CONSTRUCTION OF USER LIBRARIES WITH EDITLIB

Through the EDITLIB utility program, a user† can define a group of central processor routines or overlays to be a library. That library is available then to the system loader by specific direction in the loader control statements for a job. For example, LDSET(LIB=libname) can be used immediately before the statement calling for load from that library. For any job, routines on user libraries can replace or supplement those on the SCOPE system libraries without changing the system libraries. With EDITLIB, a user library can be modified by the addition, deletion, or replacement of routines; and statistics about library contents can be listed.

The user library must contain assembled central processor routines, programs, or text records output by the COMPASS assembler, one of the SCOPE compilers or loader generated overlays. Library records can be independent programs, subroutines, or overlays. Binary output from SEGLOAD cannot be made part of a library. In this discussion, these items will be referred to collectively as programs. Unassembled text records in BCD format, peripheral processor programs, and source language programs can not be made a part of user libraries.

EDITLIB considers each program on the user library to be a single unit occupying a SCOPE logical record. It extracts the name, entry points, and external references from tables output with the program assembly and uses them to construct tables describing the library file. Library tables are used by the loader to locate programs on the file. EDITLIB changes the tables when the user library is modified. Format of user library tables is the same as that for system libraries. A user library file created by EDITLIB will contain:

Assembled programs

Tables referring to
- Entry points
- External references
- Program numbers
- Program names

The program number table is used to link external references, entry points, and program names.

†The ensuing discussion is limited to user, as opposed to system, libraries. EDITLIB also can create and maintain SCOPE system libraries and create deadstart tapes. Its capabilities for user libraries are a subset of those for system library functions. System programmers should consult the SCOPE 3.4 System Programmer's Reference Manual for information about manipulating the running system or constructing deadstart tapes.

The library tables will accommodate a maximum of 2047 programs, with the restriction that the total programs do not have more than 2047 entry points.

The user library file generated by EDITLIB can be on mass storage or magnetic tape. If the library file name is assigned to a tape file before EDITLIB is called, the library will be in sequential format on that tape, with the library tables preceding the programs. Otherwise, the library will be in random format on mass storage. When the random library file is to be retained as a permanent file, the library file name should be associated with a permanent file device before EDITLIB is called.

A user library should not be copied to tape; although the copy operation will complete, the resulting sequential file cannot be used subsequently by the loader or modified by EDITLIB. Further, caution must be used if the copied tape is used as the source of programs in the creation of a new library. Since replacement or deletion of programs in a library is accomplished by deleting only addresses, the invalid programs will still be available and will be copied to the tape. Unless the user knows the tape structure and positions the file accordingly, EDITLIB search procedures may find and add the obsolete programs.

The user is responsible for cataloging and attaching any permanent files that will be used by EDITLIB while performing the task specified on each directive.

EDITLIB CONTROL CARD FORMAT

The EDITLIB utility routine is called by an EDITLIB card in the control card record. A parameter on this card specifies the file that contains EDITLIB directives. These directives provide details for creating or manipulating the user library.

The control card is:

```
EDITLIB(USER,I=lfndir,L=lfnlist)
```

All parameters are optional.

USER	Distinguishes user library definition from system library; default is USER.
lfndir	Logical file name containing directive record; default is INPUT. I is identical to I=INPUT.
lfnlist	Logical file name to receive listable output; default is OUTPUT. L is identical to L=OUTPUT.

When the EDITLIB control card is encountered during job processing, EDITLIB will access the next unprocessed record on the INPUT file, unless the I parameter names another source of directives.

The following deck structure assembles two program and adds them to an existing library:

```
jobcard.
COMPASS.
FTN.
EDITLIB(USER)
7/8/9 end-of-record
   COMPASS program to be assembled
7/8/9
   FORTRAN Extended program to be compiled
7/8/9
   Directives instructing EDITLIB to add programs from LGO file
6/7/8/9 end-of-file card
```

EDITLIB DIRECTIVE FORMAT

The directive record for EDITLIB must contain only valid directives. EDITLIB considers the first 72 columns of each 80 column card or 90 column card image to contain a separate directive. Blanks may be used freely; EDITLIB removes them except in a literal or comment field. Required format for directives is similar to SCOPE control card format.

EDITLIB requires parentheses around parameter lists; the format is:

keyword.   or   keyword(parameter list)

Required parameters must appear in the order given; optional parameters can appear in any order after the required parameters. Optional parameters have the format parameter=value; all others are required.

EDITLIB directives perform several functions:

Establish library limits

Create or modify library

Manipulate files or records

List statistical data

Set parameters for INTERCOM use

Add comments to output file

Directive format and use is summarized below:

`LIBRARY(libname,`$\left\{\begin{matrix}\texttt{OLD}\\\texttt{NEW}\end{matrix}\right\}$`)`	Defines library to be created or modified
`FINISH.`	Terminates library manipulation
`ENDRUN.`	Stops execution of directives
`ADD(prog,from,AL=level,FL=fl,FLO=o)`	Adds new program to library
`REPLACE(prog,from,AL=level,FL=fl,FLO=o)`	Replaces program on library
`DELETE(prog)`	Deletes program in library
`SETAL(prog,level)`	Changes access level
`SETFL(prog,fl)`	Changes field length requirements
`SETFLO(prog,`$\left\{\begin{matrix}\texttt{0}\\\texttt{1}\end{matrix}\right\}$`)`	Sets FL override bit for INTERCOM
`CONTENT(prog,lfn)`	Lists program data from file
`LISTLIB(prog,lfn)`	Lists program data from library file
`REWIND(lfn)`	Rewinds file
`SKIPF(`$\left\{\begin{matrix}\texttt{n}\\\texttt{prog}\end{matrix}\right\}$`,lfn)`	Skips ahead n records or to prog
`SKIPF(n,lfn,F)`	Skips n files forward
`SKIPB(`$\left\{\begin{matrix}\texttt{n}\\\texttt{prog}\end{matrix}\right\}$`,lfn)`	Skips back n records or to prog start
`SKIPB(n,lfn,F)`	Skips n files backward
`*/`	Inserts comments in output

The prog parameter in these directives can take several forms:

> A single program name can be stated. EDITLIB will search the entire file specified to find the named program.
>
> An asterisk can replace the program name. EDITLIB will process all programs from the current file position to end-of-file.
>
> A range of programs to be included in directive execution can be specified with a + between the first and last programs to be processed. In a file with records A,B,C,D,E, the range B+D represents B,C,D.
>
> A range of programs to be excluded from directive execution can be specified with a – between the first and last programs to be considered. In a file with record A,B,C,D,E, the range B–D represents A and E.
>
> An asterisk can replace either the first or last program named in a range. For the first named program, it is equated with the current file position; for the last, it is equivalent to end-of-file.
>
> For the ADD, REPLACE, and REWIND directives only, several individual programs can be stated. In a file with records A, B, C, D, E, the parameter D/B/E represents D and B and E. EDITLIB will search the entire file specified to find the named programs.
>
> A single program to be excluded from directive execution can be specified with a dash (–) preceding the program name or with the program name appearing at both ends of the range of programs to be excluded.

Program names must not exceed 7 characters. Any character supported by SCOPE is legal. If characters EDITLIB uses for delimiters are in a name, the entire name must be written as a literal between dollar signs. These characters are:

$ ( ) – + = . , / blank

Any dollar sign to be included in the program name must be prefixed by a second dollar sign.

The EDITLIB search procedure for programs named in directives depends on whether the file named is in library format. To find programs on files not in library format, EDITLIB reads the prefix table accompanying each assembled program from the current file position to the end-of-file, then rewinds and searches the file from its beginning if necessary. However, the file INPUT will not be searched. To find programs on a library file, the file is searched by reading the program name table and searching the table for the first and last program name. In both cases, the last named program in a range must be locatable after the first one named.

The interpretation of the * also depends on file format. The current position of a library file is always defined to be the beginning of the program name table. Current position of other files is simply the beginning of the next record on the file, which can be controlled by the user with file manipulation directives. An * replacing the last program is equivalent to stating end-of-file.

Examples of names acceptable to EDITLIB:

Parameter Format	Resulting Program Name
PROG12	PROG12
$PROG12$$$	PROG12$
$I-O$	I-O
AA BB	AABB
$AA BB$	AA BB
3AB	3AB

Library file names should not begin with ZZZZZ since these are reserved for system names.

COMMENTS IN DIRECTIVE RECORD

Comments on separate cards may be interspersed freely in the directive record. A comment is not executed but is listed in the material output by EDITLIB. To be considered a comment, columns 1 and 2 must contain the characters

```
*/
```

Any SCOPE characters may follow the slash.

CREATING A LIBRARY

DELIMITING LIBRARIES

The library to be manipulated is identified in a LIBRARY directive. Every directive record calling for library creation or modification must have at least one such directive. A FINISH directive is required to mark the end of library construction. File manipulation cards may appear between LIBRARY and FINISH.

LIBRARY must precede all other directives except comments or file manipulation directives.

$$\texttt{LIBRARY(libname,}\left\{\begin{array}{l}\texttt{OLD}\\\texttt{NEW}\end{array}\right\}\texttt{)}$$

This directive specifies the library to be created (NEW) or modified (OLD). The libname parameter is both the library name and the name of the file containing the library during this job. OLD must be given when an existing library is referenced by libname. NEW must be given when libname refers to a new library or directory.

To indicate that library manipulation is complete, the directive is:

```
FINISH.
```

An ENDRUN should follow FINISH; if ENDRUN is not supplied by the user, EDITLIB will insert it.

STOPPING EXECUTION

During directive processing EDITLIB first interprets each directive in the record excluding comment cards. Execution begins after all directives are interpreted.

When an ENDRUN is encountered during the execution phase, execution stops. In most instances ENDRUN will be the last directive in the record. By placing it earlier in the record, syntax of following directives can be checked but an error will not produce premature termination.

Format is:

```
ENDRUN.
```

ADDING PROGRAMS DURING LIBRARY CREATION

ADD directives between LIBRARY (lfn,NEW) and FINISH directives create a user library. Programs can be added from any file attached to the job, as long as the program contains the necessary prefix table material at the beginning of the assembled information.

Programs may be added to a user file from an existing library by using the optional parameter, LIB, with the ADD or REPLACE directives. It directs EDITLIB to search the program name table of a file that is in library format. The search of the PNT is important in that, when a DELETE or REPLACE is performed on a random library file, the original program on the file is not destroyed; only the pointer to it is removed.

Optional AL, FL, and FLO parameters on the ADD directive provide information primarily for use by INTERCOM. AL specifies access level, which determines whether or not a given INTERCOM user can attach and use the program named. AL also is used to mark programs for any access by control cards. Unless the rightmost bit of the binary representation of this octal parameter is set to 1, the program is available only to internal calls.

The FL parameter, meaningful only for central processor programs, indicates the central memory field length necessary to load and execute this program and any overlays or other programs it may call. INTERCOM requests the field length indicated by this parameter whenever a user calls for the program. The system routine that reads control cards also uses FL when in the reduce mode.

The FLO parameter is strictly for INTERCOM use. If the field length required for program execution exceeds the default field length normally assigned to an INTERCOM user, the FLO parameter should be set to 1. Then INTERCOM will allow the user to expand his field length to execute properly.

The SETAL, SETFL, and SETFLO directives can be used to change the values of these parameters in an existing library.

ADD directive format:

```
ADD(prog,from,AL=level,FL=fl,FLO=o,LIB)
```

prog — Name of program or range of programs to be added.

from — Logical file name where assembled program currently resides.

level — Optional access level parameter of 1-4 octal digits. If the program is to be accessed by a control card, level must be an odd number. Default is 0.

fl — Optional parameter of 1-5 octal digits specifying field length required for program execution. Default is 0. The field length used will be that specified on the job card or the last RFL card encountered.

o — Optional field length override bit. Default is 0. 0 no override allowed; 1 specified on job card may override field length specified in PNT.

LIB — Optional parameter indicating that`from` is a library name.

If AL, FL, or FLO values are wanted in the new library tables, they must be explicitly stated in the directive, even if the addition is to be made from an existing library.

Examples of valid formats and their results:

`ADD(*,TREES)` — All programs between current position and the end of file TREES will be added.

`ADD(RAINIER,MTS,FL=14400)` — All of file MTS is searched for program RAINIER; field length of 14400 is required to execute RAINIER.

`ADD(REDWOOD-SEQUOIA,TIMBER)` — All programs on file TIMBER, except REDWOOD, SEQUOIA, and all those between, are added.

`ADD(*+ASPEN,YELLOW)` — All programs from the current position of YELLOW through program ASPEN are added.

`ADD(BIG/SHARP,LEAF)` — File LEAF is search as needed and programs BIG and SHARP are added.

`ADD(ALP,LIBR,LIB)` — The program name table of library LIBR is searched for program ALP which, when located, is added to the current library.

MODIFYING EXISTING LIBRARIES

Existing user libraries in random file format are modified by the ADD, REPLACE, and DELETE directives that change programs in the library. The SETAL, SETFL, and SETFLO directives change parameters in the program name table of entries for existing libraries. These directives must be issued between the LIBRARY(lfn,OLD) and FINISH directives.

The ADD directive is the same as that used to create the library; all directive parameters are the same. If the program to be added duplicates the name of a program already in the library, an informative message will be issued; and the new record will not be added to the library.

```
ADD(prog,from,AL=level,FL=fl,FLO=o,LIB)
```

The REPLACE directive differs from the ADD directive in that it causes a program with an identical name to be deleted from the library before the new program is added. If a program with that name does not exist, an informative message will be issued; and the new program will be added to the library.

```
REPLACE(prog,from,AL=level,FL=fl,FLO=o,LIB)
```

Parameters have the same meaning as those of the ADD directive. AL, FL, and FLO values must be stated explicitly if values other than the defaults are wanted. Current values in source library or existing library tables will not be preserved when ADD or REPLACE are used.

Examples of valid formats and their results:

```
REPLACE(MAPLE,TREES,FLO=1)
```

Existing program MAPLE is deleted; program MAPLE is added from file TREES; FLO is set to 1; FL and AL are set to default values.

```
REPLACE(OAK,TREES)
```

Existing program OAK is deleted and replaced; FL, FLO, and AL receive default values.

```
REPLACE(ACORN,TREE,LIB)
```

Program name table for library TREE is searched for program ACORN. The named program is deleted from the current library and the new program ACORN is added from library TREE.

The DELETE directive logically deletes all references to the named program from library tables.

```
DELETE(prog)
```

prog Name of program to be deleted

Examples and their results:

```
DELETE(BIRCH+ASH)
```

Programs BIRCH through ASH on library being modified will be deleted.

DELETE(LAUREL-MADRONE) All programs on existing library except LAUREL, MADRONE, and those between will be deleted.

Programs named in a DELETE or REPLACE directive will be logically deleted from the library file. Records in the file will not be overwritten; but in the case of a REPLACE, the file will be extended with the addition of a new program. To completely eliminate programs from the library, it is necessary to construct a new library using the old one as the source.

Fields in tables of existing libraries are changed by the following directives:

SETAL(prog,level)

level New access level of 1-4 octal digits.

SETFL(prog,fl)

fl New field length of 0 to 377777 octal.

SETFLO(prog,o)

o New field length override parameter. 0 is the default value and does not allow override; 1 allows override.

MANIPULATING SOURCE PROGRAMS OR LIBRARY FILES

File manipulation cards may appear anywhere in the directive record. Alternately, these functions may be performed by control cards. EDITLIB issues requests to the SCOPE routine CIO to execute these directives.

REWIND(lfn) Rewinds file lfn.
or
REWIND(lfn/lfn/...lfn) Rewinds all files named.

SKIPF($\left\{\begin{array}{l}\texttt{n}\\ \texttt{prog}\end{array}\right\}$,lfn) Skips forward n decimal records or to beginning of named program on file lfn.

SKIPF(n,lfn,F) Skips forward n decimal files on multi-file lfn.

SKIPB($\left\{\begin{array}{l}\texttt{n}\\ \texttt{prog}\end{array}\right\}$,lfn) Skips backward n decimal records or to beginning of named program on file lfn.

SKIPB(n,lfn,F) Skips backward n decimal files on multi-file lfn.

Both the SKIPF and SKIPB directives result in positioning at the beginning of a file or record. Count begins with 1. When beginning-of-information or end-of-information is encountered, a skip by count is terminated. For a skip by name, the entire file is searched, if necessary, in the'direction stated. Skip by program name is applicable to sequential files only.

LISTING STATISTICS

A list of information about any or all programs on a library file or a file of assembled information is obtained by the LISTLIB and CONTENT directives. Information listed comes from the program tables output with every assembled record. It includes:

Program name

Date, time, and compilation or assembly machine

Entry points

External references

AL and FL values

Length of object deck in CM words

Type of program; relocatable or absolute

A library file is listed by:

```
LISTLIB(prog,lfn)
```

Part or all of the library can be listed depending on the number of programs indicated by the prog parameter.

Any file of assembled programs, whether in library format or not, is listed by:

```
CONTENT(prog,lfn)
```

prog	Program or range of programs to be listed
lfn	Logical file name

Information appearing on the EDITLIB output file as a result of the ADD and REPLACE directives is the same as that produced by the CONTENT directive.

USER EDITLIB EXAMPLES

Job MTCREAT creates a sequential user library on a tape.

```
MTCREAT.
REQUEST(MTLIB,LO,VSN=14444)             Requests 7-track tape to hold new library
REQUEST(SORCEFL,MT,VSN=14445)           Requests tape with assembled source
                                        programs
FTN.
EDITLIB(USER)
7/8/9
      FORTRAN Extended program to be compiled
7/8/9
LIBRARY(MTLIB,NEW)                      Initiates construction of new library MTLIB
REWIND(SORCEFL)                         Rewinds source file
REWIND(LGO)                             Rewind binary output from FORTRAN Ex-
                                        tended program
ADD(*+SHASTA,SORCEFL)                   Adds programs from beginning of file
                                        through SHASTA
SKIPF(3,SORCEFL)                        Skips 3 programs on file
ADD(HOOD,LGO)                           Adds program from LGO file
ADD(*,SORCEFL)                          Adds all remaining programs on SORCEFL
FINISH.                                 Terminates library construction
ENDRUN.                                 Stops execution
6/7/8/9
```

Job MTCHNGE modifies the library created above. Directives for EDITLIB are on tape 12000.

```
MTCHNGE.
REQUEST(MTLIB,LO,VSN=14444)
REQUEST(DIRECTS,NT,VSN=12000)
EDITLIB(I=DIRECTS)
6/7/8/9
```

Job BIRDS creates a random format library file and makes it permanent. Source files exist on permanent files GULLSPF and WRENSPF.

```
BIRDS.                                       Job card
REQUEST(BIRDLIB,*PF)                         Requests permanent file device for library
ATTACH(GULLS,GULLSPF,ID=PETERSON)            Attaches permanent file as lfn GULLS
ATTACH(WRENS,WRENSPF,ID=PETERSON)            Attaches permanent file as lfn WRENS
EDITLIB(USER)                                Calls EDITLIB
CATALOG(BIRDLIB,BIRDLIBRARY,ID=PETERSON)     Catalogs library as permanent file
7/8/9
LIBRARY(BIRDLIB,NEW)                         Establishes library name
ADD(*,GULLS)                                 Adds all files from GULLS
ADD(CACTUS-HOUSE,WRENS)                      Adds all files from WRENS except CACTUS
                                             through HOUSE
FINISH.                                      Terminates library
ENDRUN.                                      Stops execution
6/7/8/9
```

Job CHECK uses EDITLIB to check syntax of all directives, but does not execute.

```
CHECK.
EDITLIB(USER)
7/8/9
ENDRUN.                                      Stops execution here
LIBRARY(OLDLIB,OLD)
DELETE(SPARROW)
REPLACE(HAWK,INPUT,FLO=0)
SETAL(SHRIKE,777)
SETFLO(ROBIN,1)
SETFL(CREEPER,55000)
.
.
.
FINISH.
6/7/8/9
```

UTILITY PROGRAMS 8

A set of peripheral processor and central processor utility programs exists which can be called by control cards or by keyboard entries at the operator console.

Utility jobs conform to the normal deck structure. The job deck contains the following cards:

`Job card`	
`Request cards`	Equipment assignment
`Program cards`	Data operations
`6/7/8/9`	End of job

If only utility programs are to be executed, a short field may be specified on the job card. A field length of 12000 octal is adequate for all copy utilities. The programmer need not specify field length as the default values are sufficient for most jobs.

Equipment should be requested for all files which do not reside on public devices. Tapes can be rewound and positioned upon request. Each utility program is called by specifying its name. Parameters for execution of the program appear in a list after the name.

Error recovery is handled by SCOPE. If a parity error persists after a number of recovery retries, a parity message appears on the display console; and the copy operation may be abandoned by the operator because of the indeterminate state of the data.

FILE COPYING

The copy utilities discussed below expect input files to be delineated by either a logical end-of-record marker that is a SCOPE logical record terminator of level 17 or a double tape mark. Complete files can always be copied. SCOPE logical records within those files can be extracted selectively when a SCOPE logical record level is specified in the utility call. However, records within files created by Record Manager cannot be copied unless the file has record and blocking types equivalent to those of SCOPE logical records.

COPY TO END OF INFORMATION

```
COPY(file1,file2)
```

File1 is copied onto file2 until a double end-of-file or an end-of-information is detected on file1. Then both files are backspaced over the last file mark if a backspace is legal on the device. If parameters are omitted, files INPUT and OUTPUT are assumed. COPY will not operate on S, L or BCD tapes. It is useful for binary files, disk files, and SCOPE standard binary tapes.

COPY BINARY AND CODED RECORDS AND FILES

Files or SCOPE logical records in either binary or coded mode can be copied by one of the four following routines. The minimum field length for these routines is 5000 octal. When L tapes are being copied, the minimum is 2000 octal, plus the length of the largest physical record to be copied.

COPY BINARY FILE

```
COPYBF(file1,file2,n)
```

COPY CODED FILE

```
COPYCF(file1,file2,n)
```

COPY BINARY RECORD

```
COPYBR(file1,file2,n)
```

COPY CODED RECORD

```
COPYCR(file1,file2,n)
```

file1,file2 Name of the input and output files. Information is copied from file1 onto file2. If names are not specified, INPUT and OUTPUT are assumed.

n Decimal number indicating how many files or records are to be copied. If n is omitted, only one file or record is copied.

COPYBF and COPYCF terminate when the specified number of files are read, or when an end-of-information is encountered. Files are delimited by a level 17 on SCOPE tapes or tape marks on S and L tapes. These routines will not copy past EOF tape labels.

COPYBR and COPYCR terminate when the specified number of records are read or when a file mark is encountered. For example, if the card specifies 100 records but the file contains only 50 records, the copy operation terminates after 50 records. When an end of reel is detected, the next reel is obtained, label checking/writing is performed if the tape is labeled, and the function continues normally on the next reel. If an EOF is encountered on the input file before the record count is exhausted, the copy operation will cease (but not abort) at that point. A message is entered in the dayfile. An EOF is written on file2 and backspaced over.

A formatted FORTRAN write to disk may produce more than one line per logical record. When COPYCF or COPYCR is used to copy the file to an S tape, it will not detect the line images as separate records.

If an end-of-information is encountered on the input file before the record/file count is exhausted, the copy operation will cease (but not abort) at that point. A message is entered in the dayfile; an EOF is written on file2.

Although not primarily implemented for that purpose, the copy routine is capable of limited format conversion. The following matrix shows format conversion copies that can be handled successfully:

		OUTPUT		
		SCOPE	S	L
INPUT	SCOPE	Yes	Yes †	Yes
	S	Binary Yes BCD Yes †	Yes	Yes
	L	Binary Yes BCD Yes †	Yes †	Yes

† This kind of conversion cannot be guaranteed. Maximum record size for S tape output files is 512 words (5120 coded characters). Maximum physical record size for coded SCOPE tapes is 1280 characters. If these sizes are exceeded, the job is terminated.

COPY BCD FILE

```
COPYBCD(file1,file2,n)
```

This routine copies files to a magnetic tape where each line image is a discrete physical record, so the tape may be listed off line. The decimal number of files to be copied, n, are copied from file1 to file2. Only one file is copied if n is omitted.

Each line of the input file is assumed to be terminated by 12 bits of zero. The output file is written in BCD. Each line is a physical record of 148 characters, with the line terminator zeros converted to blanks.

When an EOF is encountered on the input file, a printer carriage control character for a skip to top of next page is written on the output file before an EOF is written. Thus, the final printed output begins each file at the top of a new page. Stray characters appear at the top of this page as a result of the skip and EOF marks on the tape.

Control card formats and the interpretation follow:

```
COPYBCD.                    (INPUT,OUTPUT,1)

COPYBCD(n)                  (INPUT,OUTPUT,n)

COPYBCD(file1)              (file1, OUTPUT,1)

COPYBCD(file1,n)            (file1,OUTPUT,n)

COPYBCD(file1,file2)        (file1,file2,1)

COPYBCD(file1,file2,n)      (file1,file2,n)
```

COPY SHIFTED BINARY FILE

```
COPYSBF(file1,file2)
```

A single file is copied from file1 to file2, each line is shifted right one character, and a leading blank character is added at the beginning of the line. If parameters are omitted, INPUT, OUTPUT are assumed. Neither file1 nor file2 can be a coded mode tape, since files are assumed to be binary.

This routine is used to format a print file where the first character of each line is to be printed rather than treated as a control character. A blank character is added to the beginning of each line, which will result in single line spacing when the file is printed. A minimum field length of 10000 octal is required for COPYSBF.

Example:

The following cards produce a listing of the decks on the COMPILE file produced by UPDATE.

```
LSTDECK.
UPDATE(Q)
COPYSBF(COMPILE,OUTPUT)        Compile file rewound automatically (by UPDATE)
7/8/9
*COMPILE decknames
6/7/8/9
```

COPY VALIDATION

One or more consecutive SCOPE logical records on one file may be compared with the same number of consecutive SCOPE logical records on another file to determine if they are identical. COMPARE is often used to verify a tape copy operation.

```
COMPARE(file1,file2,n,lev,e,r)
```

file1,file2 Files to be compared.

n Number (decimal) of SCOPE logical records in file1 to be compared to file2. Default value is 1.

lev End-of-record level number (octal). Default value is 0.

e Number (decimal) of non-matching word pairs to be written to the OUTPUT file for each non-matching record. Default value is 0.

r Number (decimal) of non-matching records to be processed during the comparison. Included in non-matching record OUTPUT file if e parameter is given. Default is 30000.

Comparison begins at the current position of each file and continues until the number of end-of-records of specified or higher level has been passed over. If all pairs of records are identical, the dayfile message is GOOD COMPARE; otherwise, it is BAD COMPARE. Additional information which can identify the non-matching records requires use of the e and r parameters.

Examples:

```
COMPARE(RED,BLUE)
```

Compares next SCOPE logical record on file RED with next record on file BLUE.

```
COMPARE(RED,BLUE,6)
```

Compares next six SCOPE logical records regardless of level or end-of-record marks; but each end-of-record on file RED must have the same level as the corresponding end-of-record on file BLUE for a good compare.

```
COMPARE(RED,BLUE,3,2)
```

Compares two files from their current positions to and including the third following end-of-record mark having a level number of 2 or greater. Both the SCOPE logical records and end-of-record levels must match for a good compare.

A bad comparison produces the message BAD REC.n. on the output file, where n is the record number, counting the first one read on each file as number 1. To obtain more information, errors and records must be specified as parameters.

```
COMPARE(GREEN,BLACK,3,2,5,1000)
```

This will do the same comparison as the previous example; but will list, on OUTPUT, the first five discrepancies between corresponding words in the files, together with their positions in the record. Positions are indicated in octal, counting the first word as 0. The limit of pairs of records to be read in which discrepancies are found is 1000; it is chosen as an infinitely large number greater than the number of records to be compared, because details are wanted about all discrepant records. If two long files were to be compared, for instance, 20 might be used as the records parameter, so that a reasonably large number of discrepancies would be described in detail; but if, through an error, the two files were completely different, an enormous and useless listing would not be produced. Furthermore, the comparison will be abandoned if this limit is exceeded, and the files will be left positioned where they stand.

A discrepancy between the levels of corresponding ends-of-records will be listed on OUTPUT, and the comparison will be abandoned, leaving the files positioned immediately after the unlike end-of-records.

Mode need not be specified in the COMPARE card. It is handled in the following manner, using the last example for file names. The first record of the first-named file (GREEN) is first read in the binary mode. A redundancy check is produced; the file is backspaced and re-read in coded mode. If another redundancy check occurs, the fact is noted in file OUTPUT, the corresponding record of the second-named file (BLACK) is skipped over, and the process begins again. If the coded read is successful, the corresponding record of the BLACK is read in coded mode. If this record of BLACK produces a redundancy check, the fact is noted in file OUTPUT, and nothing further is done with that record. Each record of file BLACK will be read in the same mode as that in which the corresponding record of GREEN was successfully read; but if the record of GREEN was unsuccessfully read in both modes, the record of BLACK will be read in the same mode as the preceding record of BLACK. Once a record of GREEN has been read without redundance in one mode or the other, following records of GREEN are read in the same mode until a change is forced by a redundancy deck.

Mass storage records can be read indifferently in either mode, so that the above strategy imposes no difficulty if a tape file is being compared with a mass storage file; as long as the tape file is named first on the COMPARE card.

COPY X TAPE TO SCOPE TAPE

The COPYXS routine converts binary X tapes, supported by previous versions of SCOPE, to SCOPE standard format tape. The binary X tape logical structure contains 512-word PRU's with short PRU's of sizes that are variable multiples of central memory words or 136 character PRU's.

```
COPYXS(xlfn,scplfn,n)
```

xlfn Logical file name of the input X tape

scplfn Logical file name of the output SCOPE tape

n Number of files (decimal) to be copied. Default value is 1.

The COPYXS card is to be used in the following manner:

```
REQUEST(xlfn,S)
REQUEST(scplfn,S)
COPYXS(xlfn,scplfn,n)
```

Both tapes must be requested as S format.

The output tape will be produced in SCOPE standard format, but it will be flagged in the SCOPE tables as S format. To read the output tape in the same job, the following control cards would be needed:

```
UNLOAD(scplfn)
REQUEST(scplfn,MT)
```

The COPYXS routine cannot determine when end-of-information occurs on an X tape; therefore, at least n files to be copied must exist on the X tape. Neither the input nor the output tape is rewound after conversion. After the requested number of files has been copied, the output tape is backspaced and positioned directly in front of the first tape mark preceding the SCOPE EOF trailer label. Subsequent files may be copied to the output tape; however, the block count in the trailer label will be incorrect.

FILE CONSOLIDATION (COPYN)

Files may be consolidated or merged with COPYN. SCOPE logical records from up to ten binary input files may be extracted and written on one output file. Input may be from tape, card, or mass storage files. Output may be to a tape, card, or mass storage file.

Directive cards associated with the COPYN routine determine the order of the final file. Several tapes may be merged to create a composite tape. A routine may be selected from a composite tape, temporarily written on a scratch tape, and transmitted as input to a translator, assembler, or programmer routine, eliminating the need for tape manipulation by the second program. In its most basic form, COPYN can perform a tape copy.

```
COPYN(f,outlfn,inlfnl...)
```

f — Format of output record:

zero	Copy records verbatim
non-zero	Omit ID from record

outlfn — Output file logical file name

inlfn — Input file logical file name

SCOPE logical records to be copied may or may not have an ID prefix table containing the name of the program or the name associated with the record. A record ID format consists of the first seven characters of the second word of each record. If records do not contain an ID, record identification cards must specify the record number–the position of the record from the current position of the file. Records without an ID are copied verbatim to the output file.

The format of the binary input files depends on the storage media. A binary tape file consists of the information between the load point and a double EOF; this file may contain any number of single EOF marks. A mass storage file ends at end-of-information; a card file must be terminated with EOR card.

On the output file, a file mark for an output tape is written by using a WEOF card in the desired sequence; or it may be copied in a range of records and counted as a record.

Deck structure for a COPYN job in which all input files are other than INPUT:

```
Jobcard
Request cards as necessary
COPYN call
7/8/9
COPYN directives
6/7/8/9
```

DIRECTIVE CARDS

Directive cards for COPYN use are REWIND, SKIPF, SKIPR, WEOF and record identification cards. These cards are read from INPUT when COPYN executes. The directive cards are free-field; they may contain blanks, but must include the separators indicated in each card description. The ordering of the directive cards establishes the material written on the output file. Directive cards are written on the system OUTPUT as they are read and processed. When an error occurs, the abort flag is set; and a message is printed on OUTPUT followed by the card in error. This card is not processed, but an attempt is made to process the next directive card. When the last directive card is processed, the abort flag is checked; if it is set, the job is terminated. Otherwise, control is given to the next control card.

The directives are described below:

```
REWIND(lfn)
```

The REWIND directive generates a rewind of the named file, file p which must be one of the input or output file names given on the COPYN control card. The file may not be the system INPUT file.

```
SKIPF(lfn,n)
```

With this directive, n (decimal) file marks on a tape file lfn may be skipped forward (+n) or backward (–n). Requests for other types of files will be ignored. No indication is given when SKIPF causes a tape to go beyond the double EOF or when the tape is at the load point.

`SKIPR(lfn,n)`

With this directive, n (decimal) records may be skipped forward (+n) or backward (—n) on tape file lfn. Zero length records and file marks must be included in n. Requests for other types of files will be ignored.

`WEOF(lfn)`

This directive writes a file mark on file lfn, which must be an output file named on the COPYN control card.

The record identification card contains the parameters which identify a SCOPE logical record or set of records to be copied from a given file.

`p1,p2,p3`

p1 — First record to be copied or the beginning record of a set. The name associated with the record or a number giving the position in the file may be specified.

p2 — Last record to be copied in a set of records:

- name — SCOPE logical records p1 through p2 are copied. p2 must be located between p1 and the file end.
- decimal integer — Number of records to be copied, beginning with p1. Zero length records and file marks are counted. Copying stops when the file end is encountered, even if the count has not been reached.
- * — p1 through an EOF mark are copied.
- ** — p1 through a double EOF mark are copied.
- / — p1 through a zero length record are copied.
- 0 or blank — Only p1 is copied.

p3 — Input file to be searched. If p1 is a name, and p3 is omitted, all input files declared on the COPYN card are searched until the p1 record is found. If it is not located, a diagnostic is issued. If p1 is a number and p3 is omitted, the last input file referenced is assumed. If this is the first directive card, the first input file on the COPYN card is used.

Examples of record identification cards:

`SIN,TAN,INPUTA` — Copies all SCOPE logical records from SIN through TAN from file INPUTA.

`SIN,10,INPUTA` — Copies 10 SCOPE logical records from file INPUTA, from SIN through SIN + 9.

`SIN,TAN` — Searches all input files beginning with current file or first input file. When SIN is encountered, all logical records from SIN through TAN are copied.

`SIN,,INPUTA` — Copies SCOPE logical record SIN from file INPUTA.

`1,TAN,INPUTA` — Copies the current SCOPE logical record through TAN from file INPUTA.

`1,10,INPUTA` — Copies 10 logical records, beginning with the current logical record on file INPUTA.

`1,*,INPUTA` — Copies the current logical record through the first file mark encountered on file INPUTA.

FILE POSITIONING FOR COPYN

Files manipulated during a COPYN operation are left in the position indicated by the previously executed directive. The file containing p1 will be positioned at the record following p2. Other files will remain effectively in the same position.

When COPYN is searching for a named record and p3 has been omitted, each input file is searched in turn until either the named record is found or the original position of the file is reached. The job INPUT file, however, is not searched end around.

In contrast to the end around search, a copy operation does not rewind files. An end-of-file terminates a copy even if the record named in p2 has not been encountered. Since the output file is not repositioned after a search, COPYN may be re-entered. Therefore, the programmer is responsible for any REWIND, SKIP, or WEOF requests referencing the output file.

COPYN does not check for records duplicating names on other files. If such records exist, the programmer is responsible for them. COPYN will use the first record encountered that matches the name on a directive card.

Examples of file positioning:

1. Record identification card: `REC,,INPUT1`

Input file INPUT1	ABLE	BAKER	. . .	REC	SIN	TAN	ZEE	E E O O F F

If INPUT1 were positioned at TAN, TAN and ZEE would be examined for REC. The double EOF would cause ABLE to be the next SCOPE logical record examined, continuing until REC is read and copied to the output file. INPUT1 would then be positioned at SIN.

2. Record identification card: `RECA`

Input file INPUT1, positioned at B1

A1	B1	. . .	Z1	EOF EOF

Input file INPUT2, positioned at load point

A2	RECA	D2	EOF EOF

Input file INPUT3, positioned at load point

A3	B3	C3	. . .	Z3	EOF EOF

All records from B1 through A1 are searched to find RECA; this repositions INPUT1 to B1. A2 is searched, and when RECA is found, it is copied to the output file; INPUT2 remains positioned at D2. INPUT3 is not searched.

3. Record identification cards and binary records on INPUT file. Directive cards are:

```
REC,,INPUT
JOB1,JOB3,INPUT
ABLE,,IN2
7/8/9
REC (binary)
7/8/9
JOB1 (binary)
7/8/9
JOB2 (binary)
7/8/9
JOB3 (binary)
7/8/9
```

Because the INPUT file is not searched end-around, REC and JOB1 through JOB3 must directly follow the requesting record identification cards in the order specified by them. An incorrect request for an INPUT record terminates the job.

RECORD CONSOLIDATION (COMBINE)

```
COMBINE(file1,file2,n)
```

For this operation, n(decimal) SCOPE logical records are read from file file1 and written as one SCOPE logical record (level 0) onto file file2. The file is not positioned prior to initiating this operation. If the files file1 and file2 have not been previously defined by REQUEST cards, they will be assumed to be on a public device. Files cannot be on S or L tapes. Binary input can produce binary output only.

PROGRAM REPLACEMENT (COPYL)

The COPYL utility routine allows named programs in an existing file to be replaced. The files must reside on mass storage or on SCOPE format tape. Two input files are used: the first can contain programs with a prefix table as output by the SCOPE compilers or assembler, overlays and segments, and deadstart PP programs that do not have prefix tables. The second input file must contain programs with prefix table information which are to replace programs with the same name on the first input file. The output file will duplicate the first input file using programs from the second input file as appropriate. Since the replacement programs must be identifiable by name, COPYL cannot be used to replace overlays or deadstart records.

```
COPYL(extlfn,replfn,outlfn,prog)
```

extlfn Logical file name of existing file; default name is OLDLIB

replfn Logical file name of file with new programs; default name is BINARY

outlfn Logical file name of output file; default name is NEWLIB

prog Optional name of last program on extlfn to be copied to outlfn

The logical file names and program names can contain only letters and numbers. A literal delimited by S will not be accepted.

On file replfn, programs may be in any order. When COPYL processing begins, replfn is rewound and examined to identify the programs on it. Then the file is rewound in anticipation of copy operations. Files extlfn and outlfn should be positioned by the user if necessary; they are not rewound by COPYL. Programs on extlfn are copied to the output file in the order they are encountered, substituting programs from replfn as appropriate.

The output file will have the same number of programs as the file of existing programs, unless the prog parameter is used in the COPYL call. The prog parameter will terminate processing. Otherwise, an end-of-file on extlfn terminates COPYL, with an end-of-file being written to the output file.

As programs are copied to the output file, a message identifying the program is sent to the dayfile. Any programs on file replfn which do not exist on extlfn are similary identified. Consequently, it is possible to use COPYL to obtain a listing of programs on a file by declaring that file the replacement file for a dummy input file; as in:

```
COPYL(DUMMY,GOODLFN,DUM)
```

Since no programs with matching names exist on file DUMMY, all programs on GOODLFN will be listed in the dayfile.

MULTI-FILE TAPE LISTING (LISTMF)

LISTMF requests a list of the contents of labeled multi-file set tape, mfn. The multi-file set must have existing status.

```
LISTMF(M=mfn,P=p)
```

LISTMF rewinds the multi-file set, mfn, and then positions to file p. If p is absent, a value of one is assumed. The contents of the label are extracted from the buffer and written to the OUTPUT file. Each succeeding file is positioned, specifying the previous number plus one. Files are positioned and labels are listed until end-of-multi-file-set status is returned (code 21). The set remains positioned at end-of-set. An example of multi-file tape processing appears on page 4-28.

OCTAL CORRECTION ROUTINE

The LOC utility program may be called with a control card or from the keyboard. Octal corrections must appear in a separate record on the INPUT file. They are entered in central memory of the job field length. The addresses to be modified can be cleared before corrections are made.

Control card format may be:

```
LOC.
LOC(to)
LOC(from,to)
```

to Last relative address to be cleared; clearing will start at RA.

from First relative address to be cleared.

LOC can be used to clear memory by providing an empty record in the INPUT file, as in:

```
Jobcard.
LOC.
7/8/9
6/7/8/9
```

The octal correction cards must have an address beginning in column 1; leading zeros may be dropped in the address. The data word begins in column 7; spacing in the data word is not important but the word must contain 20 digits.

PRESETTING UNUSED MEMORY

The Loader Reference Manual should be consulted for a description of how unused core memory may be preset prior to execution of a loaded program.

DEBUGGING AIDS 9

Debugging aids include DUMP and TRAP; requests for their use are submitted with normal jobs. TRAP is described in full in the LOADER Reference Manual.

DUMP

The function of the dump program is to enable a programmer to obtain a printed output when the job terminates which represents the contents of selected memory areas containing job-related information. The dump can be obtained in a variety of formats, and can be produced upon either normal or abnormal termination of the job.

TYPES OF DUMPS AND DMP CONTROL CARDS

Five types of dumps are possible: relative, control point area, exchange package, absolute, and ECS. All except the ECS dump are identified at the beginning of the printout along with the call that produced the dump.

The dump control card is written in any of three forms:

```
DMP(from,to)
DMP(to)
DMP.
```

Portions to be dumped are selected by specifying a first word address (from parameter) and a last word address (to parameter) on a DMP control card, in a console entry, or from a CP program. If the from parameter exceeds the field length, no dump results unless an absolute dump has been requested.

RELATIVE DUMP

This dump may include all or any part of the job field length (FL) from RA through the last word address (RA+FL). If the last word address exceeds FL, FL will be substituted instead.

```
DMP(x,y)
DMP(y)
```

If the from/to addresses specified by x and y are numeric, they must be expressed as octal numbers. If only y is specified, x is to be zero and the dumped area extends from RA through the address specified by y relative to RA.

Examples:

`DMP(500,6500)` Produces a dump of locations 500 octal through 6500 octal relative to RA.

`DMP(6000)` Produces a dump of the locations from 0 through 6000 octal relative to RA.

CONTROL POINT AREA DUMP

This dumps 200(octal) words of the control point area. The value specified for the x and y parameters must be equal and must not be zero.

Format: `DMP(x,y)`

Example: `DMP(1,1)`

EXCHANGE PACKAGE DUMP

The resulting dump consists of the following three parts:

The exchange jump package.

The first 100(octal) locations (RA to RA+77) of the FL.

100(octal) locations before and after the address to which the P register points, provided they are within the FL.

The following information is stored in the 16-word exchange package.

P	Program register contents
RA	Central memory address of beginning of user field length
FL	Central memory address of field length limit
EM	Error mode
RE	ECS reference address
FE	ECS field length
MA	Monitor address applicable only to machines with monitor exchange jump instructions

A0-A7	Contents of A registers 0-7
B1-B7	Contents of B registers 1-7
X0-X7	Contents of X registers 0-7

When the exchange jump package is dumped, the following information is given also if addresses are within the field length. A message **OUT OF RANGE** appears if they are outside the field length.

C(A1)-C(A7)	Contents of addresses listed in registers A1-A7
C(B1)-C(B7)	Contents of addresses listed in registers B1-B7

If the P register equals zero, the P address in bits 30-47 of RA+0 will determine the locations to be dumped.

If the P register or the P address in RA+0 is less than 200(octal), the first address dumped will be 100(octal).

If the P register and P address in RA+0 are both zero, part 3 will not be dumped.

Format:

```
DMP.
DMP(0)
DMP(0,0)
```

This dump appears without a user call when a job terminates abnormally.

ABSOLUTE DUMP

All locations in central memory may be dumped whether they are within the FL or not. The x parameter must be six octal numbers and the first must be 4, 5, 6, or 7. DMP substracts 4 from this digit to determine the absolute address to be dumped.

If the y parameter is less than 400000(octal), it is treated as y+400000. If only the y parameter is specified, it must be greater than 400000; x is assumed to be 400000. For example, DMP(400000,405000) dumps the first 5000 words in central memory. If the x parameter is greater than the central memory FL, no dump will result. If the y parameter is greater than central memory FL, it will be adjusted to avoid wrapping around central memory.

x Parameter	Maximum CM Address Dumped
4xxxxx	77 777 (32K)
5xxxxx	177 777 (65K)
6xxxxx	277 777 (98K)
7xxxxx	377 777 (131K)

Format: `DMP(x,y)`

The installation may install SCOPE such that absolute dumps cannot be made.

ECS DUMP

A dump is produced of the specified area of ECS.

Format: `DMPECS(x,y,f,lfn)`

ECS from location x to y is dumped. The dump begins at the closest multiple of 10(octal) less than or equal to x, and ends at the closest multiple of 10(octal) greater than y–1.

f Selects the print format per line according to the value:

0 or 1	4 words in octal and in display code.
2	2 words in 5 octal digit groups and in display code.
3	2 words in 4 octal digit groups and in display code.
4	2 words in octal and in display code.

lfn Specifies the dump file; if absent or zero, file OUTPUT is assumed.

DUMP FORMATS

To obtain a dump at program termination, DMP cards may be placed anywhere after the control card that executes the programs. To obtain a dump at abnormal termination, DMP control cards must follow the EXIT control card.

The standard dump option produces an octal core dump when the DMP control card is encountered. Each line of storage printed contains up to four central memory words, with the address of the first word at the beginning of the line. Printing of a central memory word is suppressed when that word is identical to the last word printed. When the next non-identical word is encountered, its address is printed and marked by a right-pointing arrow.

MODE ERRORS

The mode error shown in the exchange jump package and listed on a job dayfile classifies the error which caused job termination. As explained with the MODE control card in section 4, mode conditions can be overridden to allow the job to continue in spite of the error.

Mode errors reported from job termination are:

0 Program attempted to jump to location 0

1 Address referenced outside field length

2 Infinite operand used

3 Address out of field length, or infinite operand

4 Indefinite operand used

5 Address is out of field length or indefinite operand used

6 Infinite or indefinite operand used

7 Infinite or indefinite operand used, or program attempted to jump to location 0

10 Program attempted to use an exchange jump instruction that does not exist on the hardware

LOAD MAP

A map of central memory after programs are loaded can be obtained by the MAP control card.

```
MAP(option)
```

Option	Extent of map to be produced
ON	Full map
OFF	No map
PART	Map but omit entry point address

The map will appear on the printed output file to show items such as type of load, location of programs, common blocks and tables, and entry points. Load maps of system library program such as the compilers are never produced.

The selected option will remain in effect until another MAP card is used or the job ends. When MAP does not appear in the control cards or as a parameter on a loader control card, an installation default option is used.

For further information refer to the LOADER Reference Manual.

CHECKPOINT/RESTART 10

A job may be ended as the result of machine malfunction, operator error, or program error. Such an abnormal end may occur at any time during program execution. Valuable machine time would be lost if an abnormal end during one of the last steps of a long program required restarting the program from the first job step.

The Checkpoint/restart system facility captures the total environment of a job on magnetic tape; so the job may be restarted from a checkpoint, rather than from the beginning of the job. Total environment includes all files associated with the job. Only SCOPE standard format 7-track tape may be used. For mass storage files (drum or disk), the complete file is captured as well as the relative position within that file. For magnetic tape files, only the relative position on the tape is captured, so the tape may be properly re-positioned during restart.

Checkpoint/restart cannot handle:

Rolled-out jobs

Random files (except random permanent files)

Multi-file reels

ECS Resident files

ECS Buffered files

Checkpoint/restart will re-attach and reposition all permanent files, including random files and SIS files. Extend permission to make permanent the information written thus far should be obtained by the user before he takes a checkpoint. Jobs that request a rewrite in place may not be restartable if the last checkpoint taken does not reflect such updated files. The checkpointed program cannot recognize that file updates have already been made.

Each time a checkpoint dump is taken during job execution, a file is written containing all information needed to restart the job at that point. Each checkpoint dump is numbered automatically in ascending order by the system. A number of checkpoint dumps should be taken during a long job, so the user can return to any one of the previous checkpoints to restart his job.

CHECKPOINT DUMP TAPE

With a REQUEST or LABEL control card, the user may specify an unlabeled or labeled tape at installation default density with checkpoint disposition on which the checkpoint dumps are to be written. REQUEST or LABEL should precede the checkpoint request. If no such tape is supplied, checkpoint will define an unlabeled tape with the name CCCCCCC the first time checkpoint is requested, including operator initiated checkpoints. In any event, only one checkpoint dump tape should be defined for a job.

The following are examples of checkpoint tape definition:

```
REQUEST(CKPOINT,MT,CK)
REQUEST(CHECK,MT,CK,HI,IU)

LABEL(CKTAPE,W,L=CHECKPT001,X=CS)
```

Checkpoint/restart defines the following files for its use:

CCCCCCC
CCCCCCI
CCCCCCM
CCCCCCO

The user should refrain from using these file names. User SCOPE logical records should not have a level 16, since checkpoint uses level 16 for internal processing.

CHECKPOINT REQUESTS

Checkpoint dumps may be requested by control cards placed in the job stream where the dumps are desired.

Control card format is:

```
CKP.
```

Checkpoint dumps, however, are more often requested by an executing program or command entered by the operator. An executing program would request checkpoint at various logical points, such as end-of-file, x logical records processed, x seconds of elapsed time, etc. Checkpoint requests may be issued more than once. The request takes the following form:

```
CHECKPT param,sp
```

sp Mass storage files to be processed. 0 all files; non-zero numeric for files in param list. Assumed 0 if sp is not given.

param Address of a parameter list; format follows:

59	17	11 0
	n	0000
lfn1	f1	
lfn2	f2	
lfnn	fn	

n Defines number of lfn entries in following list, to a maximum of 42 (decimal).

lfn Name of user mass storage files to be processed: left justified display code.

f Octal number indicating specific manner in which lfn is to be processed.

0 Mass storage file is copied from beginning-of-information to its position at checkpoint time, and only that portion will be available at restart. The file is positioned at the latter point.

1 Mass storage file is copied from its position at checkpoint time to end-of-information, and only that portion will be available at restart. The file is positioned at the former point.

2 Mass storage file is copied from beginning-of-information to end-of-information; the entire file will be available at restart time. The file will be positioned at the point at which the checkpoint was taken.

3 The last operation on the file determines how the mass storage file is copied.

For a general call to checkpoint using the CHECKPT macro, the sp field is zero. Also:

If n = 0, all mass storage files assigned to the job, including INPUT, OUTPUT, PUNCH, PUNCHB, and LGO, will be copied to the checkpoint dump tape in the manner determined by the last code/status (f flags).

If n ≠ 0, all mass storage files named in the lfn list will be copied to the checkpoint dump tape in the manner determined by the f flags; however system mass storage files will be copied as determined by the last operation performed on each file.

If the value of the sp field is non-zero in the macro call, only the lfn's supplied by the user in the param list, plus system files will be processed. Processing is determined by the f flag settings.

When the manner of copying a mass storage file is to be determined from the last operation on the file, checkpoint derives f values from the last code status as follows:

f = 0 if code/status ends in 4, 5, 6, or 7.

f = 1 if code/status ends in 0, 1, 2, or 3 and end-of-information bit is set.

f = 2 if code/status ends in 0, 1, 2, or 3 and end-of-information bit is not set.

Generally, these values cause the entire mass storage file to be copied for: write operations, read operations resulting in end-of-information status, and rewind operations (excluding some OPEN functions).

The checkpoint macro generates the following code:

Form in X0

59			39	35	23	17 0
CKP	1	1		sp		param

```
RJ  SYS=
```

CHECKPOINT EXAMPLES

All operator or CKP control card requests are processed in the same manner as the COMPASS example.

FOR COMPASS USERS:

```
         CHECKPT   PARAM
         .
         .
PARAM    DATA    0
```

All mass storage files would be processed (sp=0, n=0) at checkpoint.

FOR FORTRAN (RUN) USERS:

```
DATA variable/0/
.
.
.
CALL CHEKPTR (variable)
```

or

```
variable=0
CALL CHEKPTR (variable)
```

FOR FORTRAN EXTENDED USERS:

```
DATA variable/0/
.
.
.
CALL CHEKPTX (variable)
```

or

```
variable=0
CALL CKEKPTX (variable)
```

For the preceding examples, checkpoint processing is performed for all mass storage files as described in the first example where sp=0, n=0.

Selected files may be processed in the manner shown in the following example:

```
DIMENSION KPARAM (4)
KPARAM (1)=30000B
KPARAM(2)=5LTAPE1.OR.10000B
KPARAM(3)=6LTAPE23.OR.10000B
KPARAM(4)=5LTAPE3
.
.
.
CALL CHEKPTR(KPARAM,1)         or       CALL CHEKPTX(KPARAM,1)
```

The user should rewind overlay files prior to requesting checkpoint if they are on mass storage.

FORTRAN overlay programs should declare the mass storage overlay files to be TAPEn files and use REWINDn before CALL CHEKPTR (variable) or CALL CHEKPTX (variable).

EXAMPLE FOR FORTRAN (RUN):

```
OVERLAY(TAPE9,0,0)
PROGRAM MAIN(...,TAPE9)
DATA FILE/5LTAPE9/
.
.
CALL OVERLAY(FILE,1,0)
.
.
END

OVERLAY(TAPE9,1,0)
PROGRAM OVER1
DATA PARAM/0B/
.
.
REWIND9
CALL CHEKPTR(PARAM)
.
.
END
```

RESTART REQUEST

The RESTART control card directs a job to be restarted from its checkpoint tape. All parameters are optional and order independent.

```
RESTART(name,n,S=xxx)
```

name — Name of checkpoint file as defined at checkpoint time. Default name is CCCCCCC.

n — Number (decimal) of checkpoint to be restarted. If this parameter is omitted, a default value of 1 is assigned by restart. If it is greater than the number of the last checkpoint taken, the restart attempt will be terminated.

xxx — xxx is the number (decimal) of words in the smallest physical record on any S tape attached to the job. If this parameter is omitted, standard SCOPE PRU size (512) is assumed.

If any S tapes with small physical records are attached to the job and the S parameter is not set to reflect this, the tapes may not be positioned correctly.

After locating the proper dump on the checkpoint tape, the restart program requests all tape files defined at checkpoint time, and repositions these files. Then, all mass storage files are requested, specifying the type of device on which the file resided at checkpoint time. The operator may assign the file to a more convenient device. Files are copied from the checkpoint tape and repositioned. Restart also restores the central memory field length of the job and restarts the user's program.

The restart job should not contain a REQUEST control card or a LABEL card requesting the file name given on the RESTART card unless the checkpoint dump tape is a labeled tape or needs any parameters other than MT or CK. If no checkpoint tape is requested, restart will issue a request for the checkpoint file named on the RESTART control card, and it will use the installation default density.

If any permanent files are attached to the control point when a checkpoint is called, they are attached and positioned as they were at the time of the checkpoint.

Any direct ECS direct access user area attached to the job will be copied in its entirety to the checkpoint tape. At restart time, it will be recopied to ECS from the checkpoint file. On the job card for the restart job, the user must request at least as much ECS as was attached to the original. If reconfiguration results in insufficient ECS available to the user, restart is not possible.

A checkpoint dump may not be restarted in the following cases:

A tape file necessary for restarting the program was overwritten after the checkpoint dump was taken.

A machine error propagated bad results but did not cause abnormal termination until after another checkpoint dump.

COMMUNICATION AREAS 11

FILE ENVIRONMENT TABLE

The file environment table (FET) is a communication area supplied by the user within his field length. Any file to be written, read, or otherwise manipulated or positioned, must have its own FET. The FET is interrogated and updated by the system and user during file processing.

COMPASS programmers can create an FET in two ways:

Use the FET creating macros FILEB, FILEC, RFILEB, or RFILEC.

Use other COMPASS instructions to build a table in the format expected by the system.

Compiler language programmmers need not be concerned with FET construction or manipulation, since the compilers will perform these tasks in response to compiler language instructions. When Record Manager is used for input/output, the user need supply only the file information table data. Record Manager will construct and manipulate the FET from information in its File Information Table (FIT). The FIT is fully described in the Record Manager Reference Manual and the System Programmers Reference Manual.

A minimum size FET is five words, which allows for processing of sequential unlabeled files. Random or labeled files, or files in which the user will process file conditions or errors with OWNCODE routines, require a longer table. Extensions to the FET–areas identified by pointers within the FET–are required for extended error and label processing. Some compilers append an area past word 13 of the FET, as explained in the respective manuals.

When S and L tapes are processed, the FET must be at least seven words in length. Word 6 is required for blocking/deblocking of files. Words 7 and 8 are required for indexed files. Word 9 is used for user OWNCODE routines. Words 10-13 are present when the LABEL macro is used.

The format of the FET is shown in figure 11-1. Some fields are pertinent only to Record Manager manipulation; a discussion exists in the reference manual for Record Manager. Other fields contain different data depending on the file mode or residence.

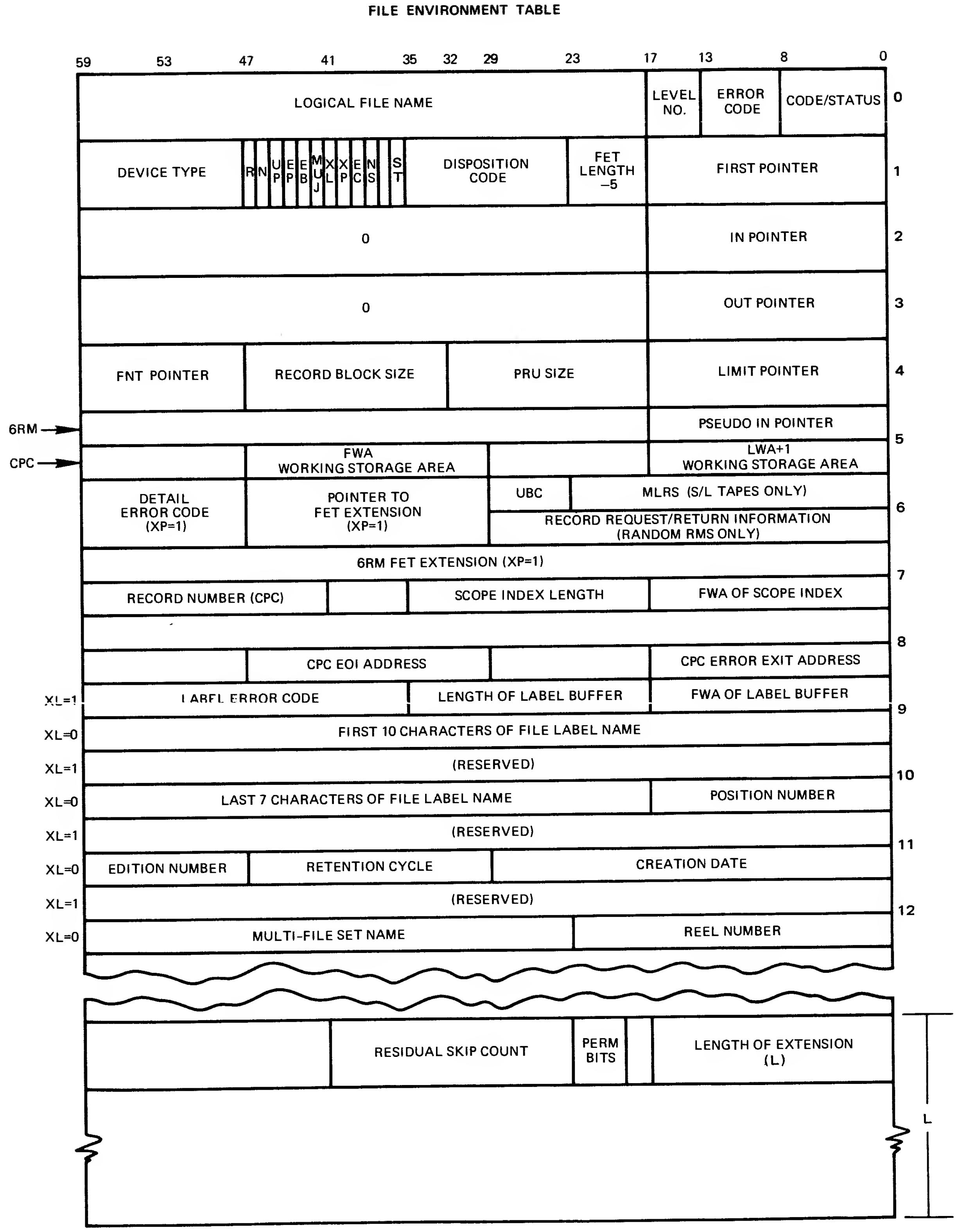


Figure 11-1. File Environment Table

FET CREATION MACROS

System macros in the COMPASS language facilitate generation of the FET.

All parameters except lfn, fwa, and f are optional. The fwa and f parameters must be in the order shown; others may be in any order. The macro parameters WSA, OWN, XPA and IND are not order dependent, but order is fixed within these parameters.

The user must specifically allocate the circular buffer location in the field length. The macro identifies but does not create the buffers.

If an FET creating macro is followed immediately by a USE statement naming another labeled common block, the FET becomes the last item in a labeled common block. The FET creation macro may not truncate properly to the minimum size required; a truncated common block may result.

Four macros are available, depending on whether the file is coded or binary, random or sequential.

CODED SEQUENTIAL FILE

```
lfn    FILEC    fwa,f,(WSA=addrw,lw),(OWN=eoi,err),LBL,UPR,EPR,XPR,
                (XPA=xpadr,lx),UBC=ubc,MLR=mlrs
```

BINARY SEQUENTIAL FILE

```
lfn    FILEB    fwa,f,(WSA=addrw,lw),(OWN=eoi,err),LBL,UPR,EPR,XPR,
                (XPA=xpadr,lx),UBC=ubc,MLR=mlrs
```

CODED RANDOM FILE

```
lfn    RFILEC    fwa,f,(WSA=addrw,lw),(IND=addri,li),(OWN=eoi,err),LBL,
                 UPR,EPR,XPR,(XPA=xpadr,lx)
```

BINARY RANDOM FILE

```
lfn    RFILEB    fwa,f,(WSA=addrw,lw),(IND=addre,li),(OWN=eoi,err),LBL,
                 UPR,EPR,XPR,(XPA=xpadr,lx)
```

Further explanation of parameter usage appears with descriptions of the FET fields below.

lfn Logical file name

fwa Circular buffer address; substituted in FIRST, IN, and OUT

f Length of circular buffer; fwa+f is substituted in LIMIT to make buffer address lwa+1; f should be at least one word larger than PRU size of the device on which the file resides

WSA Working storage area keyword; parameters required for READIN and WRITOUT; relieves user of responsibility for buffer manipulation

addrw First word address of working storage area

lw Length of working storage; when coded files are being processed, the length must be at least as long as the longest record, or data will be lost

IND Index buffer parameter keyword; required for random files only

addri First word address of index buffer

li Length of index buffer; for numbered SCOPE indexed files, length should allow one word for each record plus a one word header; for named SCOPE indexed files, two words are required for each record in addition to the index header

OWN OWNCODE routine parameters keyword

eoi Address of routine to be executed if end-of-reel, end-of-pack, or end-of-information occurs; UPR must be used

error Address of routine to be executed if file action errors occur; EPR must be used

UPR User specifies processing at end-of-reel, end-of-pack, or end-of-information; sets bit 45 of word 2

LBL Label information will follow for magnetic tape file; LABEL macro providing label information must immediately follow the FET creating macro to which it pertains

EPR User specifies handling of file action error conditions; sets bit 44 of word 2; does not set extended error processing flag

UBC Unused bit count keyword; required only for S and L tapes

ubc Specifies number of bits in last word of record that do not contain valid data

MLR Maximum record size keyword; required only for S and L tapes

mlrs Maximum number of 60-bit words in record

XPR Extended error information to be returned by system

XPA Extended label processing keyword

xpadr First word address of FET extension for extended error processing

lx Length of FET extension for extended error processing; must be 1

Examples:

To create a minimum FET for the standard INPUT file:

```
LBUFFER  EQU    65
INPUT    FILEC  BUFFER,LBUFFER
```

To create an FET for a binary random file:

```
LBUFFER  EQU    65
LINDEX   EQU    25
FILEABC  RFILEB BUFFER,LBUFFER,(IND=INDEX,LINDEX)
```

To create an FET for a labeled tape file with user processing at end-of-reel condition. OWNCODE routine is supplied:

```
LBUFA   EQU    65
TAPE1   FILEB  BUFA,LBUFA,LBL,UPR,(OWN=PROCEOR)
TAPE1   LABEL  SORTINPUTTAPE,32,90
```

To create an FET for a list file. OWNCODE routines are supplied and the working storage area is used:

```
LBUFB   EQU    65
PRINT   FILEC  BUFB,LBUFB,(WSA=LINE,14),(OWN=ENDING,ERRORS),UPR,EPR
```

FET FIELD DESCRIPTION

Words of the FET are numbered 1-13 in decimal, corresponding to the addresses lfn through lfn+12 decimal. All parameter values are octal unless otherwise noted. Bits are numbered 0-59 right to left in decimal.

LOGICAL FILE NAME (lfn) (bits 18-59 at lfn)

The lfn field contains one to seven display-coded letters or numbers starting with a letter, left justified; if less than seven are declared, unused characters are zero-filled. This field is used as common reference point by the central processor program and the peripheral processor input/output routines.

The lfn parameter declared in an FET creation macro is also used as the location symbol associated with the first word of the FET. A reference to lfn in the file action requests is a reference to the base address of the FET.

CODE AND STATUS (CS)(bits 0-17 at lfn)

The CS field is used for communication of requested functions and resulting status between the central processor program and the peripheral processor input/output routines. This field is set to the request code by CPC when a file action macro request is encountered. When the FET is generated, bits 2-17 should be zero.

The code and status bits have the following significance:

Bits 14-17 Record level number. On skip and write record requests, this subfield is set by CPC as part of the function code. On read requests, it is set by CIO as part of the status when an end-of-record is read. Initially the level subfield is set to zero when the FET is generated.

Bits 9-13 Status information upon request completion. Zero indicates normal completion. Nonzero indicates an abnormal condition, not necessarily an error; an OWNCODE routine, if present, will be executed. Status codes are described with the EOI OWNCODE and Error Exit Address discussions. Initially, this subfield is set to zero when the FET is generated.

Bits 0-8 Used primarily to pass function codes to a peripheral processor. Function codes are even numbers (bit 0 has a zero value). They are listed as CIO codes below.

When the request has been processed, bit 0 is set to one. When the FET is generated, bit 0 must be set to one to indicate the file is not busy.

Bit 0 Current status of request (0 = file being processed, 1 = request complete).

Bit 1 Specifies the mode of the file (0 = coded, 1 = binary). Bit 1 is not altered by CPC when a request is issued.

Bits 2-8 Pass function codes to a peripheral processor (file action requests).

Bits 3 and 4 These bits will be set to binary 10 if end-of-record is read, or to binary 11 if end-of-file is read.

CIO CODES

Function codes listed below can be set in the CS field by the user before calling CIO to carry out the function. They are set by CPC when file action macros are used. All values are octal.

All codes indicated by – are illegal; all reserved codes are illegal. All codes are shown for coded mode operations; add 2 for binary mode. Example: 010 is coded READ, 012 is binary READ. Upon completion of operation, code/status in FET is changed to an odd number, usually by adding 1 to the code. In some cases, code is further modified to indicate manner in which operation concluded. Example: a READ function 010, at completion, becomes 011 (buffer full), 021 (end of SCOPE logical record), or 031 (end-of-file).

000	RPHR†	054	–	130	CLOSE/NR
004	WPHR†	060	UNLOAD	134	–
010	READ	064	–	140	OPEN
014	WRITE	070	–	144	OPEN/WRITE
020	READSKP	074	–	150	CLOSE
024	WRITER††	100	OPEN/NR	154	–
030	–	104	OPEN/WRITE/NR	160	OPEN
034	WRITEF	110	POSMF	164	–
040	BKSP	114	EVICT	170	CLOSE/UNLOAD
044	BKSPRU	120	OPEN/NR	174	CLOSE/RETURN
050	REWIND	124	–		

† Applies to SCOPE format tapes only.

†† When a WRITER function is issued with level 17 specified, SCOPE changes the function to a WRITEF. Thus, a function issued as a 24 will return as a 34.

200 Series for special reads or writes (reverse, skip, non-stop, rewrite, etc.)

214	REWRITE	230	–	254	–
220	–	234	REWRITEF	260	READN†††
224	REWRITER	240	SKIPF	264	WRITEN†††
		244	–	270	–
		250	READNS	274	–

††† Applies to S and L tapes only.

300 Series used for OPEN and CLOSE

300	OPEN/NR	324	–	360	–
304	–	330	CLOSER	364	–
310	–	334	–	370	CLOSER/UNLOAD
314	–	340	OPEN	374	–
320	–	350	CLOSER		
		354	–		

400 Series reserved for CDC

500 Series reserved for installations

600 Series:

600	–	630	–	654	–
604	–	634	–	660	–
610	–	640	SKIPB	664	–
614	–	644	–	670	–
620	–	650	–	674	–
624	–				

700 Series reserved for CDC

DEVICE TYPE (dt)(bits 48-59 at lfn + 1)

The device type value will be returned to the FET device type field when a file action request is issued, if FET length exceeds the minimum. The 6-bit device type will occupy bits 54-59; bits 48-53 will hold recording technique identification for magnetic tapes, if applicable. The mnemonic is used in the REQUEST card only.

Group I

Mnemonic	Device Type	Device
AA	01	6603-I disk**
AB	02	6638 disk
AC	04	6603-II disk
AL	05	821 data file
AM	06	841 multiple disk drive
AP	07	3234/854 disk pack drive
AF	10	814 disk file
AD	12	3637/865 drum
–	13-17	CDC reserved
AX	20	ECS resident files
–	21-27	CDC reserved
–	30-37	Reserved for installations, mass storage only

** Basic 6603 with or without field option 10098 (disk speedup) installed; 6603-II is a 6603 with both field options 10098 and 10124 (speedup augment) installed.

Group II

Mnemonic	Device Type (Octal)		Recording Technique (Right 6 bits of FET dt field in binary)	
MT	40	7-track magnetic tape	xxxx00	HI density 556 bpi
			xxxx01	LO density 200 bpi
			xxxx10	HY density 800 bpi
			xxxx11	CDC reserved
			xx00xx	Unlabeled
			xx01xx	SCOPE standard U and Z labels
			xx10xx	3000 series label (Y)
			xx11xx	CDC reserved
			00xxxx	SCOPE standard data format
			01xxxx	CDC reserved
			10xxxx	S data format
			11xxxx	L data format
NT	41	9-track magnetic tape	xxxx10	HD density 800 cpi
			xxxx11	PE density 1600 cpi
			xxxx00	CDC reserved
			xxxx01	CDC reserved
			xx00xx	Unlabeled
			xx01xx	SCOPE standard U label (ANSI)
			xx10xx	3000 series label (Y)
			xx11xx	CDC reserved
			10xxxx	S data format
			00xxxx	SCOPE standard data format
			01xxxx	CDC reserved
			11xxxx	CDC reserved

Mnemonic	Device Type (Octal)		Recording Technique (Right 6 bits of FET dt field in binary)
** _	42	member multi-file set 7-track tape	Same as in MT
** _	43	member multi-file set 9-track tape	Same as in NT
** _	62	7-track multi-file set tape	Same as in MT
** _	63	9-track multi-file set tape	Same as in NT

** Code is generated when a tape is declared to have MF characteristies; the multi-file set code 62 or 63 is used only in system tables; it is not returned to the user's FET.

Group III

Mne- monic	Device Type (Octal)	Device
* TR	44	Paper tape reader
* TP	45	Paper tape punch
–	46-47	Reserved for installations
LP	50	501,512 line printer
L1	51	501 line printer
L2	52	512 line printer
–	53-55	CDC reserved
–	56-57	Reserved for installations
CR	60	405 card reader
KB	61	Remote terminal keyboard
–	64-65	Reserved for CDC
–	66-67	Reserved for installations
CP	70	415 card punch
DS	71	6612 keyboard/display console
* GC	72	252-2 graphic console
* HC	73	253-2 hard copy recorder
* FM	74	254-2 microfilm recorder
* PL	75	Plotter
–	76-77	Reserved for installations

* Codes are defined but supporting software is not provided by SCOPE.

RANDOM ACCESS (R) (bit 47 at lfn + 1)

A one in the R field indicates a random access file. R may be set to 1 by using the RFILEB or RFILEC macro. When a file is opened or closed, the R setting determines action performed with regard to the SCOPE index as shown below:

OPEN	FET R=0	FET R=1
No SCOPE index	No index action	FET Rbit is set to zero and a non-fatal diagnostic is written to the dayfile.
SCOPE index	No index action	Index is read into index buffer; if index buffer is not specified,FET R bit is set to zero and a non-fatal diagnostic is sent to dayfile.

If a non-existent file is opened, the value of the R bit is not altered. Only files on allocatable devices may have an index. The FET R bit is set to zero if the file is on a non-allocatable device.

CLOSE	FET R=0	FET R=1
File had SCOPE index on last OPEN	File is flagged as not having index	If index buffer exists, the index is written; and file is flagged as having SCOPE index. If buffer is not specified, non-fatal diagnostic occurs.
File had no SCOPE index on last OPEN	No index action	If index buffer is specified, index is written; and file is written; and file is flagged as having a SCOPE index. If index buffer is not specified, a non-fatal diagnostic occurs.

The above actions will be performed only if the contents have been altered since the file was last opened.

When any other file action request is issued, the r setting determines the access method to be used. If r=0, the file will be read or written beginning at the current location. If r=1, the file will be read or rewritten according to the logical disk address in FET word 7, or written at the end-of-information; and the logical disk address returned to FET word 7.

RELEASE (N) (bit 46 at lfn + 1)

When this bit is set and a mass storage file is read sequentially, mass storage allocated to the file will be released as the file is read. Backward positioning combined with sequential file reading may yield unpredictable results.

USER PROCESSING (UP) (bit 45 at lfn + 1)

The UP bit may be used to control tape end-of-reel processing and sequential disk pack end-of-pack processing. If the UP bit is zero, unit swapping is automatic without notification to the user; the function in process when end-of-reel or end-of-pack is detected will be completed on the next unit. If the UP bit is set to one, the user will be notified when an end-of-reel or end-of-pack condition arises. End-of-reel for tape files is defined as a tape mark followed by an EOV1 label for labeled tapes and SCOPE format unlabeled tapes, or as the first tape mark after the EOT reflective spot for unlabeled S and L tapes.

If the UP bit is set, end-of-reel status and end-of-pack status (02) are returned in bits 9-13 of the FET code and status field. Functions that do not transfer data from the circular buffer will have been completed; data transfer functions may be re-issued as indicated by an examination of the buffer pointers. If CPC is in use, control will return to the EOI OWNCODE routine, if declared in bits 30-47 of lfn + 8. Then the user must terminate processing. If a continuation reel or pack is desired, a CLOSER function should be issued.

ERROR PROCESSING (EP) (bit 44 at lfn + 1)

The EP bit is set when the calling program is to be notified of error conditions arising from file actions. Error codes returned to the code and status field are listed under the error exit address field (page 11-21). Control is given to the user OWNCODE routine at error address when EP is set. If EP has not been set, the operator is informed of the error and must authorize job termination or continuance regardless of the error.

NO RECOVERY (NR) (bit 43 at lfn + 1)

This bit may be set to control error recovery. If it is set, no attempt will be made to recover errors encountered while reading data on magnetic tape.

MULTI-USER JOB (MUJ) (bit 42 at lfn + 1)

Set only when the file is being processed by a multi-user job. Currently, the EDITOR routine in INTERCOM is the only multi-user job. When bit 42 is set, user id, user table addresses, and a special code (for routine 3TT) appear in lfn + 5.

EXTENDED LABEL PROCESSING (XL) (bit 41 at lfn + 1)

This bit affects processing of labels on magnetic tape. Format to be used in the label fields in lfn + 10 through lfn + 12 depends on this setting. Standard label processing of required labels occurs when XL=0. If XL=1, the user can process optional labels, as described under Tape Label Processing.

EXTENDED ERROR PROCESSING (XP) (bit 40 at lfn + 1)

Extended error processing has been added to the SCOPE 3.4 system. The error processing available under the last version of SCOPE remains unchanged; codes are still returned through bits 9-13 of FET word 1. In addition, the upper 12 bits of FET word 7 are now used to more closely detail errors if the XP bit equals 1, as explained under FET Extension Pointer field. An error message is displayed on the B display and is written to the dayfile. If this bit is not set, the operator is informed of unrecovered errors and has the option of dropping or continuing the job.

The EP bit must be set before control can return to the user OWNCODE to process these errors. Also, the UP bit must be set to gain control at end-of-volume.

When XP is set, the FET extension pointer in word 7 must be set.

EC (bit 39 at lfn + 1) Reserved for SCOPE.

NON-STANDARD LABEL (NS) (bit 38 at lfn + 1)

Setting this bit to 1 indicates non-standard labels exist. All processing must be done by user program.

(bit 37 at lfn + 1) Reserved for SCOPE.

STATION (ST) (bit 36 at lfn + 1) Setting of this bit indicates the FET is for a 6000 station control point.

DISPOSITION CODE (bits 24-35 at lfn + 1)

The value shown below will be returned to the FET disposition code field when a file action request is issued, if FET length is greater than the minimum. A file with the specified default name automatically will be assigned the corresponding disposition code value at job completion.

The user cannot alter file disposition by changing this field. Rather, the DISPOSE card or macro must be used.

For FR, FL, HR, HL, and PT, SCOPE recognizes the code and its value, but does not provide routines to process them in the release system. All other codes are reserved to the system.

Code	Value (octal)	Disposition	Default File Name
CK	xx01	Checkpoint	—
IU	xx02	Inhibit automatic unload of tape	—
CI	xx03	Checkpoint and inhibit unload tape	—
SV	xx04	Inform operator to save tape	—
CS	xx05	Checkpoint and save tape	—
PU	xx10	Punch Hollerith	PUNCH
PB	xx12	Punch Binary	PUNCHB
P8	xx14	Punch 80 Column	P80C —
FR	xx20	Film Print	FILMPR
FL	xx22	Film Plot	FILMPL
HR	xx24	Hard Copy Print	HARDPR
HL	xx26	Hard Copy Plot	HARDPL
PT	xx30	Plot	PLOT
PR	xx40	Print (501,512)	OUTPUT
P1	xx41	Print (501 only)	—
P2	xx42	Print (512 only)	—
PE	xx44	Print (512 with 95 character train)	
—	xx7x	Reserved to Installation	—
—	1xxx	INTERCOM file	—
—	2xxx	INTERCOM batch	—
—	4xxx	EXPORT file	—

Codes CK, IU, CI, SV, and CS are applicable to tape files only; they may appear on LABEL or REQUEST cards. The remainder are applicable to mass storage residence only.

LENGTH OF FET (lgth) (bits 18-24 at lfn + 1)

The system FET length is determined as follows: FET first word address + 5 + lgth = last word address + 1. The minimum FET length is five words (lgth=0). If the minimum FET is used, only the logical file name, code and status field, FIRST, IN, OUT, and LIMIT are relevant. No other field will be set or checked by SCOPE. An FET of six words (lgth+1) is used if a working storage area is needed for blocking/deblocking. An FET of eight words (lgth+3) is used if the r bit is set, indicating an indexed file. Length is nine words (lgth=4), if OWNCODE routines are declared.

FNT POINTER (bits 48-59 at lfn + 4)

The FNT pointer is set by SCOPE, upon return from a file action request, to the location of the file entry in the file name table. The pointer is placed in the FET to minimize table search time and does not affect the program. The pointer will not be set if a minimum FET is used.

PHYSICAL RECORD UNIT SIZE (PRU) (bits 18-33 at lfn + 4)

The physical record unit size of the device to which the file is assigned is returned in this field when a file is opened. It is given as the number of central memory words. The PRU size is used by CPC to determine when to issue a physical read or write. PRU size will not be returned if a minimum FET is used.

RECORD BLOCK SIZE (bits 34-47 at lfn + 4)

If the file resides on an allocatable device, the size of the device record block is returned in this field when the file is opened. It is given as the number of physical record units in a record block. If the number of PRU's is not defined or is variable, the field is set to zero. Record block size is not returned if a minimum FET is used.

FIRST, IN, OUT, LIMIT (bits 0-17 at lfn + 1 through lfn + 4)

The fields contain the beginning address (FIRST) and last word address + 1 (LIMIT) which define the file circular buffer. The IN and OUT pointers indicate the address of data placed into or removed from the buffer. System and programmer use of these fields is discussed under the heading Circular Buffer Use.

WORKING STORAGE AREA (WSA) (lfn + 5)

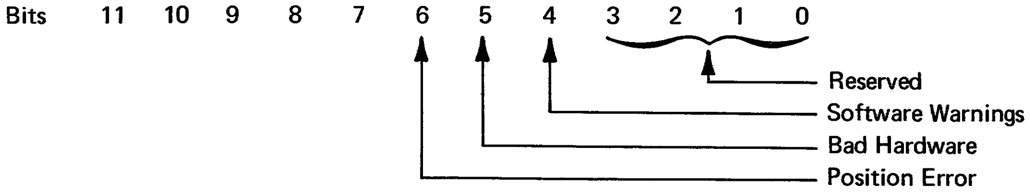
The two fields in this word of the FET specify the first word address (bits 30-47) and last word address + 1 (bits 0-17) of a secondary buffer within the program field length. The area is needed to use the system macros READIN and WRITOUT, which blocks or deblocks records from the area into the circular buffer. READIN and WRITOUT relieve the user of responsibility for circular buffer pointer manipulation.

DETAIL ERROR CODES (bits 48-59 at lfn + 6)

When the XP bit is set to 1, this field contains extended tape error processing codes which give additional detail of abnormal conditions resulting from the last input/output operation.

Codes 0-3777 are device dependent error conditions. Tape error codes are defined below. All other codes in this range, as well as those 4000-5777 are reserved for SCOPE. Codes 6000-7777 are reserved for installations.

The references to system noise record and last good record refer to procedures the system follows in recovery attempts. These are explained in the Systems Programmer's Reference Manual.



0020	25 consecutive feet of tape have been erased.
0021	Installation defined erase limit reached.
0022	Blank tape read.
0023	Incomplete erasure of bad spot on tape. Prior data may still exist.
0024	Read opposite mode successful.
0025	Noise in interrecord gap.
0026	Function not complete.
0027	Possible record fragment.

0030	Record length exceeds PRU size or MLRS size
0040	Lost data
0041	Tape parity error
0050	MMTC memory parity error
0051	Transmission parity error
0100	Position uncertain; valid data probably destroyed; a CLOSE may work
0101	Position uncertain; valid data probably intact; CLOSE will probably work

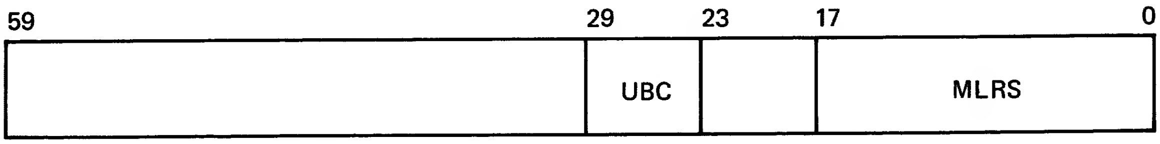
FET EXTENSION POINTER (bits 30-47 at lfn + 6)

When the XP bit is set, pointer is the required address of an FET extension. Currently, the extension is limited to a single word, but the length (L) parameter anticipates future expansion.

UNUSED BIT COUNT (UBC) (bits 24-29 of lfn + 6)

The unused bit count field is used only for files in S or L tape format. (If the device type is not magnetic tape, this word will contain indexing information). It is used for communication between the peripheral processor input/output routines and the user program.

For magnetic tapes with S or L data format, the structure of the word at lfn + 6 is:



For a READ or READSKP function, SCOPE will store into this field the number of low-order unused bits in the last data word of the record. The UBC field is not used during a READN request. For a WRITE, WRITER or WRITEF function, SCOPE will read the contents of UBC and adjust the length of the record accordingly.

For example, to write a single record of 164 decimal characters, the data length is 17, to the next highest CM word. The number of low-order unused bits in the last word would be 36. The user would set UBC = 36, set IN and OUT pointers to reflect 17 words of data, and then issue a WRITE or a WRITER.

SCOPE does not use the UBC field during a WRITEN request. UBC may range from 0 to 59, but will always be a multiple of 12 when set as a result of a read operation. If it is not a multiple of 12 for a write request, SCOPE will truncate the value to the nearest multiple of 12; if UBC is 18, SCOPE will execute as though it were 12, and if UBC is 6, SCOPE will execute as though it were 0. The field in the FET remains unchanged.

MAXIMUM LOGICAL RECORD SIZE (MLRS) (bits 0-23 of lfn + 6)

The MLRS field is applicable only when magnetic tape files in S or L format are considered. It defines the size of the largest physical record to be encountered when the S or L tape format is used. The size is given in number of central memory words.

For S tape format, if MLRS = 0, the value of the maximum PRU is assumed to be 512 words. For L tape format, if MLRS = 0, the assumed maximum PRU is LIMIT – FIRST – 1 for standard reads, and LIMIT - FIRST - 2 for READN.

Since S and L tapes record size is defined in characters, instead of central memory words, the last word may contain invalid data. Consequently, UBC is required to attest to the validity of all characters in this word.

RECORD REQUEST/RETURN INFORMATION (bits 0-29 of lfn + 6)

If the file resides on a mass storage device and has the r bit set in word 2, indexing information appears in words 7 and 8 for communication between the peripheral processor input/output routines and the user program.

For mass storage random files, the format of word lfn + 6 is:

59	29 0
	Record Request/ Return Information

The record request/return information field is set to zero when the FET is generated. Both the indexing functions and the peripheral processor input/output routines set the field during random file processing.

For other than the SCOPE indexing method, the following information is pertinent. At the start of writing a new SCOPE logical record, if the random access bit and the record request/return information field are non-zero, the latter field is assumed to contain the address of a location within an index. The PP routine inserts into that location (in bits 0-23) the PRU ordinal (starting from 1) of the SCOPE logical record. To read the record again, the random access bit should be set to non-zero and the PRU ordinal should be entered in the FET in the record request/return information field.

RECORD NUMBER (bits 36-59 at lfn + 7)

When a SCOPE indexed file is processed, this field contains the ordinal of a record identified in the index. Records are numbered beginning with 1.

SCOPE INDEX LENGTH (bits 18-35 at lfn + 7)

When a SCOPE indexed file is processed, this field contains the number of words in the index. One word for each numbered record, or two words for each named record, plus a one-word header is required.

SCOPE INDEX ADDRESS (bits 0-17 at lfn + 7)

This field contains the address of the index for a SCOPE name or number index file.

EOI OWNCODE ADDRESS (bits 30-47 of lfn + 8)

This field contains the address of a user supplied OWNCODE routine to be entered when end-of-information, end-of-reel, or end-of-pack status is encountered during magnetic tape or sequential pack processing. The UP bit must be set.

CPC enters this routine when bits 9-13 of the codes and status field is:

01	End-of-information encountered after forward operation
02	End-of-reel reached during magnetic tape forward operation
02	End-of-pack reached during sequential pack operation

Just before entering an end-of-information OWNCODE routine, CPC zeros bits 9 and 10 of the first word of the FET. However, as the routine is entered, register X1 still contains the first word of the FET as it appeared before those two bits were zeroed.

ERROR EXIT ADDRESS (bits 30-40 of lfn + 8)

This field specifies an address to receive control if an error condition occurs after a file action request. The EP bit must be set to cause control to pass to this OWNCODE address. The FET code and status field will reflect the error condition. If processing can continue, the error routine should exit through its entry point; otherwise, an abort request may be issued. If the error address field is zero, the run continues normally. The FET code and status bits reflect the error condition upon normal return to the program. Bits 9-13 of this field may be:

04 Irrecoverable parity error on last operation; or lost data on write.

10 During a magnetic tape read, the physical record size exceeded circular buffer or maximum allowable PRU size (MLRS for S and L tapes). During a mass storage write, all mass storage space meeting the file requirements was in use or otherwise unavailable.

20 Blank tape read.

21 End of multi-file set. File position number is greater than that of the last member in the set. Any subsequent attempt to reference the logical file name assigned to the nonexistent member will result in a fatal error.

22 Fatal error.

23 Index full.

24 Reserved for future use.

25 Attempt made to read or write record number n of a random file, but n exceeds index size.

26 Attempt made to read named record from random file, but name does not appear in index.

27 Attempt made to write named record on random file, but name does not appear in index, and index is full.

30 Function legal but not defined on device.

31 Permanent file permission not granted.

32 Function legal except for permanent files.

33-37 Reserved for future use.

If both EOI and error routine execution are needed, the error routine is executed. Just before entering an error OWNCODE routine, CPC zeros bits 11-13 of the first word of the FET. However, as the routine is entered, register X1 contains the first word of the FET as it appeared before those bits were zeroed.

LABEL PARAMETERS (lfn + 9 through lfn + 12)

Words 10-13 of the FET may contain information pertaining to magnetic tape labels. The format and use of these fields depends on the setting of the extended label processing bit in word 2. The LABEL macro generates fields for normal label processing. Further details appear under the Tape Label Processing heading.

Parameters in these fields must be display code. If other than the LABEL macro is used to create them, display code zero may be used to add leading zeros to numeric fields. Character fields, which are left justified, may be display code blank filled.

RESIDUAL SKIP COUNT (RSC) (bits 24-41 at P + 0)

When XP is set and P is the address of the FET extension word, RSC is the residual skip count. If SKIPF, SKIPB, or READSKP functions do not complete the specified number of skips, the count of records yet to be skipped is returned here. RSC will have a value when SKIPB encounters beginning-of-information even when the UP bit is not set. If SKIPF terminates at end-of-volume because UP is set, RSC will be set.

PERM BITS (bits 20-23 of P + 0)

The setting of these bits will duplicate that of the permanent file permission bits in the file name table. Permission is granted when the bit indicated is set.

20	Read permission
21	Extend permission
22	Modify permission
23	Control permission

These bits are set when the user issues an OPEN function.

EXTENSION LENGTH (bits 0-17 at P + 0)

The length of the extension, including word P, is required. For SCOPE 3.4, this value must be 1.

CIRCULAR BUFFER USE

For each file, the user must provide one buffer, of any length greater than a PRU size. The buffer is called circular because it is filled and emptied as if it were a cylindrical surface in which the highest addressed location is immediately followed by the lowest. The FET fields FIRST, IN, OUT and LIMIT control movement of data to and from the circular buffer.

Data is transmitted in physical record units; their size is determined by the hardware device. For example, the 6603 disk has an inherent PRU size of 64 CM words; SCOPE binary mode magnetic tape files are assigned a PRU size of 512 words.

FIRST and LIMIT never vary during an I/O operation; they permanently indicate buffer limits to the user and SCOPE.

The program that puts data into the buffer varies IN, and the program that takes it out varies OUT. During reading, SCOPE varies IN as it fills the buffer; and the user varies OUT as he removes data from the buffer. During writing, the user varies IN as he fills the buffer with data; and the system varies OUT as it removes data from the buffer and writes it out.

The user cannot vary IN or OUT automatically except when using READIN and WRITOUT functions. To change these pointers within the program a new value is inserted into lfn + 2 (IN) or lfn + 3 (OUT). For convenience, the words containing IN and OUT contain no other items, eliminating the need for a masking operation. The system dynamically checks the values of IN and OUT during data transfers, making continuous read or write possible.

If IN = OUT, the buffer is empty; this is the initial condition. If IN>OUT, the area from OUT to IN − 1 contains available data. If OUT>IN, the area from OUT to LIMIT − 1 contains the first part of the available data, and the area from FIRST to IN − 1 contains the balance.

To begin buffering, a READ function may be issued. SCOPE will put one or more PRU's of data into the buffer beginning at IN, resetting IN to one more than the address of the last word filled after each PRU is read. Data may be processed from the buffer beginning with the word at OUT, and going as far as necessary, but not beyond IN − 1. The user must then set OUT to one more than the address of the last word taken from the buffer. He sets OUT = IN to indicate that the buffer is empty.

When a READ macro request is issued, if the buffer is inactive and a read is not in process, CPC determines how much free space is in the buffer. If OUT>IN, OUT − IN words are free. If IN>OUT, (LIMIT − IN) + (OUT − FIRST) words are free. The system subtracts 1 from the number of free words, because it never must fill the last word since it would result in IN=OUT and give a false empty buffer condition. If the number of free words minus 1 is less than the PRU size, CPC does not issue a physical read request; control is returned normally.

The example below illustrates the use of IN and OUT pointers. Speed of operation is not considered; simultaneous processing and physical I/O are not attempted.

The initial buffer pointer position is:

FIRST = BCBUF
IN = BCBUF
OUT = BCBUF
LIMIT = BCBUF + 500

The user issues a READ with recall request. Ignoring the possibilities of an end-of-record or end-of-file, the system reads as many PRU's as possible (if PRU size is 64 words, 7 x 64 = 448 words) and leaves the pointers:

FIRST = BCBUF
IN = BCBUF + 448
OUT = BCBUF
LIMIT = BCBUF + 500

The user is processing items of 110 words. He takes four items from the buffer, leaving the pointers:

FIRST = BCBUF
IN = BCBUF + 448
OUT = BCBUF + 440
LIMIT = BCBUF + 500

The user issues another READ request since the buffer does not contain a complete item. The system is aware that IN>OUT, so that vacant space is LIMIT – IN + OUT – FIRST = 492 words; since it must not fill the last word, it must read fewer than 492 words.

The nearest lower multiple of 64 is 7 x 64 = 448, so the system reads 52 words into IN through LIMIT – 1, and then 396 more words into FIRST through FIRST + 395. It then resets IN so that the pointers look like:

FIRST = BCBUF
IN = BCBUF + 396
OUT = BCBUF + 440
LIMIT = BCBUF + 500

The system has just used the circular feature of the buffer; now the user must do so. The next time he wants an item, he takes the first 60 words from OUT through LIMIT – 1, and the remaining 50 from FIRST through FIRST + 49. Then he resets OUT, making the pointers:

FIRST = BCBUF
IN = BCBUF + 396
OUT = BCBUF + 50
LIMIT = BCBUF + 500

On input, this can continue indefinitely, with OUT following IN, around the buffer. The system stops on encountering an end-of-record or end-of-file, and sets the code and status bits accordingly. The system may, or may not, have read data before the end-of-record; so it is up to the user to examine the pointers and/or process the data before taking end-of-record or end-of-file action.

In writing, the process is similar, but the roles are reversed. The user puts information into the buffer and resets IN; and when he calls the system, it removes information from the buffer and resets OUT. For writing, the system removes data in physical record units and empties the buffer if possible. The user must be careful not to overfill the buffer; IN must not become equal to OUT. During the process of emptying the buffer, SCOPE resets OUT after each PRU has been written and checked for errors.

ESTABLISHING OWNCODE ROUTINES

The EOI address and error address fields in word 9 of the FET define user supplied routines. CPC calls these routines when the UP or EP bits are set.

An OWNCODE routine should be set up like a closed subroutine with execution beginning in the second word of the routine. CPC calls an OWNCODE routine by copying the exit word of CPC into the first word of the OWNCODE routine, putting the contents of the first word of the FET into register X1, and branching to the second word of the OWNCODE routine.

Termination of an OWNCODE routine by a branch to its first word causes a branch to the point in the program to which CPC would have returned if the OWNCODE routine had not been called.

Although CPC clears status bits in the first word of the FET before the OWNCODE routine is called, the contents of this word can be examined in register X1. All registers used in the main program except A1, X1, A6, and X6 are saved and restored by CPC.

TAPE LABEL PROCESSING

The label processing that occurs for magnetic tapes is indicated by the XL bit setting, bit 41 of the second word of the FET. Extended label processing is possible only when this bit is set. An explicit open is required.

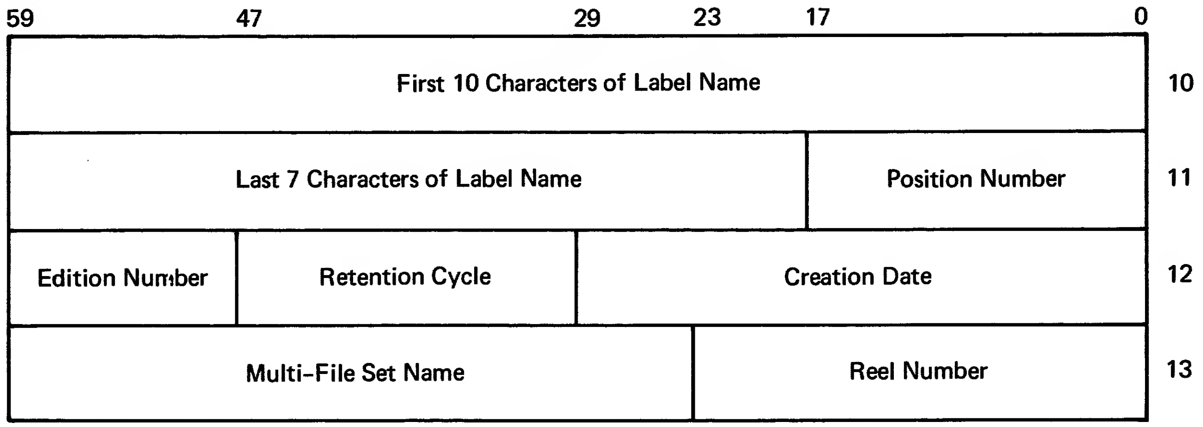
When the bit is off, the system generates output data and checks input data only for required ANSI, Z format, and Y (3000 series) format labels. Labels that are processed by standard processing (excluding Y labels) are label types VOL1, HDR1, EOF1, and EOV1. Default values are written if the user does not specify otherwise.

Checking of the VOL1 label of ANSI or Z formats ensures that the VSN requested for the job is the one assigned.

STANDARD LABEL PROCESSING

Only standard labels are processed when the XL bit is off. Any existing optional labels will be ignored.

If the FET for the file is at least 13 words long, words 10-13 hold file header label data in the following format:



When input tapes are read, any user information in these fields is compared with that written in the HDR1 label on the tape before the file is opened. A discrepancy in a label field stops job processing until the operator takes action to continue it. This checking cannot be done with FET's with less than 13 words, but any labeled tape will be accepted for processing. After the file is opened, the HDR1 label on the tape is delivered to the circular buffer for the file, as long as space exists in the buffer.

When output tapes are opened, any information in words 10-13 is used to create the HDR1 label for the file. Otherwise, default values are written. If two OPEN functions with rewind are performed, the system retains the information written the first time. Thus, a label area supplies the label information regardless of which programs run afterwards.

LABEL MACRO FOR FET FIELDS

Fields in words 10-13 of the FET can be set for standard label processing by means of the LABEL macro. This macro must follow immediately the macro creating the FET to which it pertains. The LABEL macro generates data for a file header label but does not directly cause any action of the file.

```
lfn    LABEL    labname,ed,ret,create,reel,mfn,pos
```

lfn — Logical file name used in FET creating macro.

labname — Label name or file identification of 1-17 characters; default is 17 blank characters.

ed — Edition number specifying file version of 1-2 decimal digits; default is 01.

ret — Retention indicator indicating the 1-3 digit decimal number of days the file is to be protected against accidental destruction; default is installation parameter.

create — Creation date in format of 2 digits for year and 3 digits for day(yyddd); default is current date.

reel — 1-4 decimal digits indicating sequential reel number of a multi-reel set; default is 0001.

mfn — Multi-file name of 1-6 characters indicating the set to which lfn belongs; default is binary zero.

pos — Position number of 1-3 decimal digits indicating position of file lfn in multi-file set mfn; default is 000.

The macro expansion results in display code values with binary zero fill for all parameters given. If a parameter is absent from the macro, it is binary zero filled. Character fields are left justified; numeric fields are right justified.

When a file header label is written subsequently using the FET fields, default values are assigned for any field containing binary zero. On the tape, character fields are display code blank filled and numeric fields are display code zero filled. The fields, as written on the tape, are returned to the FET.

When the information in the FET is used to check existing labels, binary zero fill characters will be converted to the display code blank appropriate for character fields or display code zero for numeric fields before comparison is made. Fields in the FET containing all binary zeros are not compared. Checking procedures compare fields in the FET with those on the tape; not all fields in the FET need be specified, neither must the FET contain a value for all fields written on the tape.

If the header label on the tape mounted does not match the FET fields, job control is at the operator option. He can attempt to locate the correct tape and assign it to the job, or accept the mounted tape with non-matching label fields. If he accepts the first tape, the values returned to the FET will reflect the header label on that tape.

EXTENDED LABEL PROCESSING

When the XL bit is set, a user label buffer, rather than the FET, is used to hold labels for processing. The system processes the required labels, and the user may process optional labels in the buffer.

Buffer location must be defined in word 10 of the file FET as follows:

	35	17 0	
Error Return Code	Length of Label Buffer	FWA of Label Buffer	10

Within the buffer, each label must be preceded by a status word.

59	47	35	23	11 0
				Characters in Label

Only bits 0-11 should be set by the user to show the number of characters in the label. Remaining fields are set and used by the label processor. The last label should be followed by the status word containing zeros in bits 0-11.

Each label in the buffer appears, in display code, with the same format it has on the tape. Appendix E lists specific label field characteristics.

When input tapes are read, the label processor searches the buffer for a HDR1 label. Any information in the label in the buffer is compared with that on the tape, with differences requiring operator action. The system validates only the HDR1 label; other labels are the user's responsibility. After an OPEN function is issued, all labels read by the system are delivered to the buffer, beginning with VOL1.

When output tapes are generated, any user labels to be written must be present in the label buffer when an OPEN or CLOSE function is issued. The buffer may, but need not, include the system required labels. The operating system will generate the required labels if they are not present in the label buffer. VOL1 labels in the label buffer will be ignored; HDR1 labels in the label buffer will be used if they are appropriate at that point in file processing. EOF1 or EOV1 labels in the label buffer will be used if they are present when the CLOSE function is issued.

For multi-file set processing with the XL bit set and calls to the COMPASS macro POSMF, word 10 of the FET must point to a label buffer. One of the first entries in the buffer must be a formatted HDR1 label with the multi-file name in the set identifier field. The position number field in the label has 4 digits; a position number of 9999 is required to write a label. Labels are always written at the end of all existing files in the multi-file set.

RA COMMUNICATION AREA

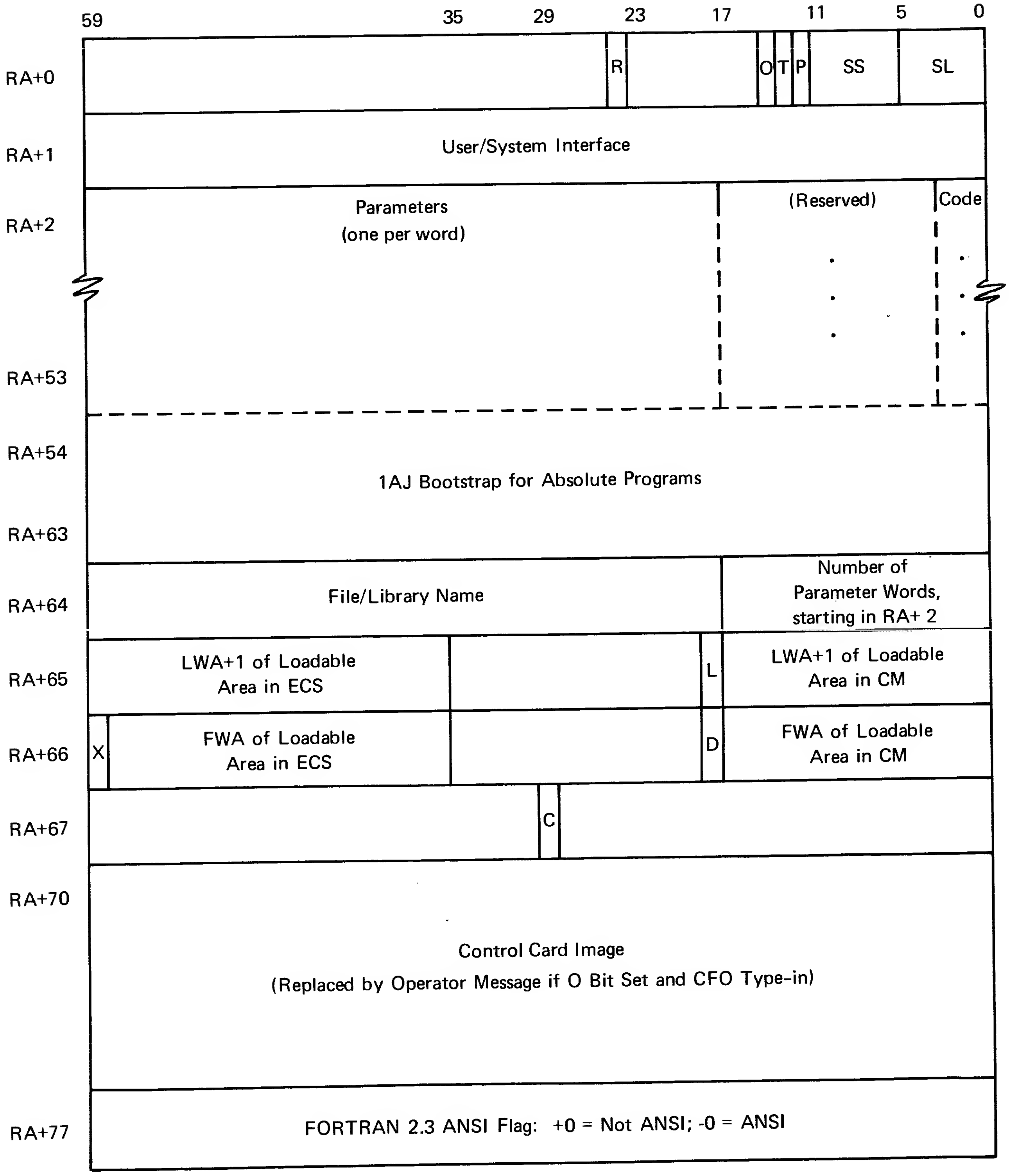


Figure 11-2. Communication Area RA through RA+100

LOCATIONS RA THROUGH RA + 100

The first 100 octal locations within a user field length are used for communication between SCOPE and a user job. Many of the words are applicable only to internal SCOPE working, and can be ignored by the programmer. Several of the fields in this area are useful in COMPASS programming when macros discussed in section 12 are called.

Figure 11-2 shows the contents of this area. The System Programmer's Reference Manual provides an explanation and use of all fields.

R Dependent job string recheck bit

O CFO flag (1 = accept comment from operator)

T Storage move flag (1 = move being attempted)

P Pause flag; when set, program will halt until the operator takes action and clears the flag with GO command; if MESSAGE is called when P is set, the message will flash on the B display

SS Sense switches 1-6 set by SWITCH cards or by operator command ONSWn

SL Sense lights 1-6 used by FORTRAN programs

L Library/file flag (1 = name is library name)

X If set, system has an XJ instruction available for use in COMPASS

C LOAD complete flag set when load requested by LOADREQ is finished

D DIS RSS flag

When a control card is read in response to CONTRLC macro, parameters will be placed in RA + 2 through RA + 53. The total number of words containing parameters will exist in bits 0-17 of RA + 64.

Codes accompanying the parameters are:

00	Continuation	04	(	10	;
01	,	05	+	16	other
02	=	06	-	17	. or)
03	/	07	blank		

The control card currently being processed exists in RA + 70 through RA + 76.

Location RA + 1 is set by the user, or macros called by the user, when a function is requested of Monitor (discussed in section 12).

COMPASS INTERFACE WITH SCOPE 12

USER/SYSTEM COMMUNICATION

A user program can request action by another part of the SCOPE operating system in several ways:

A request for PP program execution or system action can be placed in location RA+1 of the user field length to communicate directly with Monitor.

Central processor subroutine CPC (central program control) can be called through a return jump instruction. CPC will then communicate with Monitor.

A file action macro can be called. This results in a call to CPC which posts a request in RA+1 to communicate with Monitor.

The system communication routine SYS= can be called through various macros.

A Record Manager macro can be called. These macros can be used in place of direct or indirect calls to CPC.

These requests are necessary to perform all file action such as opening, closing, reading, or writing a file, in addition to receiving information such as current time or date from the system.

BASIC COMMUNICATION: RA + 1 REQUESTS

All requests from the user program to the system are made through RA+1 of the user program, which is initialized to zero. The system Monitor frequently examines RA+1 during program execution. If RA+1 is not zero, Monitor assumes that the contents are a request for a PP program or a system action, and initiates request processing. When Monitor processing is complete, RA+1 is reset to zero.

The requests to Monitor must be in the general format:

Bit 42-59	3 display code characters of a PP program.
Bit 40	1 if automatic recall is requested. With automatic recall, control is not returned to the calling program until the request is executed. If automatic recall is not requested, the user program must determine whether or not the request is complete by checking a status word.
Bit 36-39	Zero.
Bit 0-35	Parameters that are required by the particular function being requested.

The user has the option of setting RA+1 directly, or calling a system or file action macro that will set it for him. If he sets it directly, the format must conform to that shown above.

When Monitor accepts the request, it fills location RA + 1 with zeros. For all requests except RCL, TIM, ABT, or END, the zero means only that Monitor has accepted the request and has no relation to whether the requested task is complete. Normally, a user program posts an RA + 1 request, then loops until that location is zero, before proceeding with other code. The user should make sure that RA + 1 is clear before issuing a request.

Task completion is noted by the change of bit 0 in a status word from 0 to 1. For requests made with automatic recall, the complete status bit is always set to 1 before control returns to the program, as explained below. Bits 0-17 of the RA+1 request points to the status word. For file action requests, this status word is the first word of the FET for that file.

RECALL CONCEPT

A recall request issued in a program causes the central processor assigned to that job to be relinquished temporarily. The length of time that the job leaves the processor depends on whether periodic or automatic recall was requested. The amount of elapsed time before the job is reassigned the central processor is also dependent on the relative priority of the job in the system, but jobs in recall are among the first considered by the scheduling routines in SCOPE.

Periodic recall puts the job in recall status for the time Monitor requires to accept the recall request and return the job status to that of waiting for the central processor.

Automatic recall (auto recall) causes the job to relinquish control of the central processor for the time required to execute a requested peripheral processor or Monitor function. The job remains in recall until after Monitor detects a status bit change to a word in the user field length which is set when the peripheral processor completes its task. For file action requests, the complete bit is bit 0 of the first word of the FET for the file.

With programs using recall whenever appropriate, overall central processor use is improved. If a program cannot proceed until a requested task is complete, it can allow Monitor to assign the central processor to another job until such time as the task is complete. Recall is particularly useful when input/output tasks are considered.

A programmer can request recall in four ways:

RCL request to Monitor through program location RA+1

PP program call in RA+1 with recall bit set

RECALL macro request

File action macro with recall parameter

Central processor programs can post an RA + 1 request with the display code characters RCL in bits 42-59 and obtain periodic or auto recall depending on the remainder of the request. Periodic recall results from RCL in bits 42-59 with bits 0-41 containing all zeros. Automatic recall is obtained with bit 40 set to 1 and bits 0-17 containing an address of a word in the user field length which has 0 in bit 0. A PP program is expected to set bit 0 of the parameter word to 1 when its task is complete. If bit 0 is set to 1 when the RA + 1 request is posted, Monitor will cause the job to terminate with a message AUTO RECALL ERROR.

Central processor programs can request automatic recall when a PP program request is posted in RA+1 by setting bit 40 to 1.

The RECALL macro results in periodic recall when no parameter list is given with the macro. If a file name is specified, automatic recall is produced. No separate status word is involved with periodic recall. The user program must check the code and status field of the FET for complete status to determine whether program execution can continue.

When file action macros are used, automatic recall is requested by a recall parameter. Any non-blank character or string of characters may appear as this parameter. The characters RECALL are often used, but a single arbitrary character is sufficient.

The recall parameter can be specified for all the read and write macros except READIN and WRITOUT. However, the internal execution of these two macros ensures that automatic recall is always in effect.

USING CPC

Before CPC can honor a file action request, the file environment table (FET) must have been established for the file to be processed. Calling sequences to CPC may be generated either directly or through the use of system macro statements.

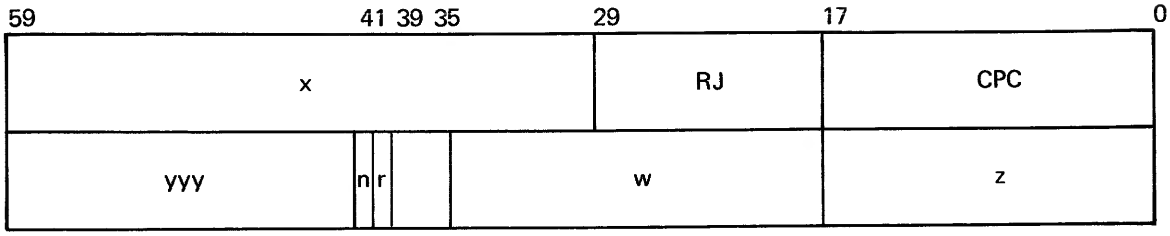
The user communicates with CPC through macro requests and the FET. Communication with SCOPE is handled by CPC through setting and checking RA+1. CPC may also cause the execution of one or more user OWNCODE subroutines for which addresses are specified in word 9 of the FET.

A normal exit is made from CPC if the request is honored and no error condition occurs. Register X1 contains zero upon exit. If the status is other than request completed, register X1 will contain the code and status bits set in the FET before the OWNCODE routine was entered.

Automatic recall should be used when the program makes an I/O or system action request but cannot proceed until that request is satisfied. SCOPE will not return control to the program until that request is satisfied. Periodic recall can be used when the program is waiting for any one of several requests to be satisfied. In this case, SCOPE will activate the program periodically so that the user can determine whether or not the program can proceed.

CALLING SEQUENCE TO CPC

Format of the calling sequence to the central program control subroutine:



RJ Return jump instruction

CPC Entry point to the CPC subroutine

r Set if auto recall requested

If n=0 indicating a file action request

yyy Display code characters CIO, or

000001 Only file recall is desired. Display code characters RCL are generated in RA+1.

000002 For most read or write functions. A function in progress will not be reissued by CPC. When the file becomes inactive, CPC issues the next request. Display code characters CIO are generated.

000003 For all other functions. When the file becomes inactive, CPC issues the request. Display code characters CIO are generated.

000004
or
000007 Equivalent to 000003; included only for compatibility with previous systems.

x SA1 base address of FET

z Request code (one of the CIO codes listed in section 11).

w Skip count for SKIPF, SKIPB, and BKSPRU; otherwise ignored.

If n = 1 indicating other than a file action request:

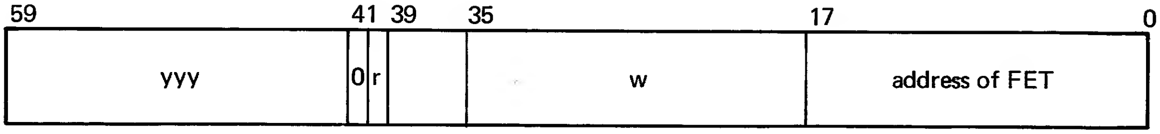
yyy Display-coded name of the called PP program

x Not relevant

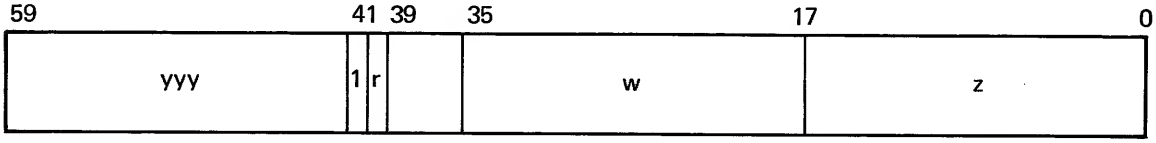
z, w Parameters as required

For file action requests, CPC places the CIO function request code in the code and status field of the FET before writing the request in RA + 1.

A file action request to Monitor is formatted by CPC in RA + 1 as follows:



A system action request to Monitor is formatted in RA + 1 as follows:



Bits not specified in the calling sequence are reserved for future system use.

CPC EXECUTION

Bit 41 of word 2 is set to 1 in the calling sequence of all requests except file action requests. This bit is actually a flag for CPC and has no relevance to either Monitor or the processing PP program. The setting of bit 41 causes CPC to recognize that the address given in A1 is not relevant, and that the word following the return jump to CPC contains a properly formatted request. No additional processing is done on these requests, except for MESSAGE. The request is simply placed in RA + 1.

A request which utilizes an FET is signalled by a value of zero in bit 41 of word 2 of the calling sequence. CPC will, in this case, do considerable processing for the user. The processing basically consists of three steps: wait until the FET is inactive; process any abnormal conditions; and initiate the new request. The high order 18 bits of word 2 in the calling sequence may contain a numerical value rather than a PP program name. These values are of the form 2X + Y, where X represents the ordinal in a table of PP program names, and Y is 1 or 0 to indicate whether or not the FET must be inactive before processing can continue. If a PP program name appears in these 18 bits, CPC waits for inactive FET status before initiating the new request.

1. Upon receipt of a file action request, CPC will wait for previous activity on the specified FET to be completed unless the Y bit is zero; CPC requests automatic recall until FET word 1 contains an odd value. The Y bit is zero for READ, WRITE, and OPEN requests. If the request is OPEN, the assumption is made that no previous activity has occurred. READ and WRITE are handled specially.

2. If the Y bit is one, the results of the previous operation will be tested. A zero in bits 9-13 of the FET code and status field indicates there are no abnormal conditions and processing goes to step 3. However, if there are abnormal conditions but no OWNCODE addresses are given, the contents of FET word 1 are saved for subsequent use as an exit parameter before processing goes to step 3. The error OWNCODE routine will be entered if bits 9-13 have a value of 4 or higher (end-of-information or end-of-reel may also be present); the EOI OWNCODE will be entered if the value is less than four. An OWNCODE routine is entered as though a return jump instruction was issued. Execution begins at the start address plus 1. An exit from the routine will, however, return control to the main program, not to CPC, which indicates that the request which triggered this activity has not been issued; and the program must decide whether to re-issue it. An OWNCODE routine is entered with X1 containing word 1 of the FET complete with bits indicating abnormal conditions; FET word 1 itself has been cleared of the abnormal bits.

3. If the new request is for READ, WRITE or REWRITE, and the FET is already active with the same request, CPC exits; it would be pointless to stop the I/O merely to reactivate it. If, however, the FET is inactive or active with a different request, steps 1 and 2 above will be executed as a subroutine. If the new request is a READ, an additional check will be made for end-of-record or end-of-file status on the previous request; the new READ will be ignored and an exit taken from CPC if either status is present. If a program is reading without recall, the user is forced to clear the EOR bit at the end of each record to ensure that he is aware of the end-of-record.

CPC now will make preparations to communicate the new request to the system. The new request code from word 2 of the calling sequence is inserted into bits 0-17 of FET word 1; the old mode bit (bit 1) is not disturbed. The RA+1 request is formatted from the following items:

PP program name obtained from the CPC calling sequence.

Setting of the auto-recall bit in the calling sequence.

First word address of the FET.

RA+1 is set and CPC waits for a zero quantity to re-appear. If the auto-recall bit was set, CPC executes step 2 above as a subroutine. CPC then exits with X1 containing zero if no abnormal conditions were encountered; otherwise X1 will contain the value from FET word 1.

CPC saves and restores all registers except A1, A6, X1 and X6.

RECORD MANAGER REQUESTS

Record Manager (RM) is a group of routines that provides input/output facilities common to all CYBER 70 systems. The COBOL, FORTRAN, and SORT compilers use Record Manager for internal input/output operations. User programs written in COBOL or FORTRAN can communicate with the Record Manager through compiler language calls; COMPASS programmers communicate through the macros listed for Record Manager under SCOPE 3.4.

Four types of file organizations are supported by RM under SCOPE 3.4:

Sequential files in physical order

Word addressable files on mass storage with continuous non-blocked data

Indexed sequential files similar to SIS version 1.0 files released under SCOPE 3.3

Direct access files containing records in fixed length blocks; record locations are determined by hashing a key to identify blocks

All four of the above are considered by SCOPE to be sequential files. None of them have SCOPE indexes similar to those discussed elsewhere in this manual.

The record and block formats supported by the Record Manager for 6000 files only are listed below.

Record Type	Description
F	Fixed length records
D	Record length is given as a character count, in decimal, by a length field contained within the record
R	Record terminated by a record mark character specified by the user
T	Record consists of a fixed length header followed by a variable number of fixed length trailers—header contains a trailer count field in decimal
U	Record length is defined by the user
W	Record length is contained in a control word prefixed to the record by the SCOPE operating system
Z	Record is terminated by a 12-bit zero byte in the low order byte position of a 60-bit word
S	Record consists of zero or more blocks of a fixed size followed by a terminating block of less than the fixed size. These S records are equivalent to the SCOPE logical records discussed elsewhere in this manual.

Block Type	Description
K	All blocks contain a fixed number of records; the last block can be shorter
C	All blocks contain a fixed number of characters; last block can be shorter
E	All blocks contain an integral number of records; block sizes may vary up to a fixed maximum number of characters
I	A control word is prefixed to each block by the operating system.

RECORD MANAGER MACROS

COMPASS 3.0 macros used for Record Manager reside in the system text overlay IOTEXT; if system defaults are installed, macros also reside in overlay SYSTEXT. General macro names and functions are given below; specific variants of these macros are detailed in the Record Manager Reference Manual. Run-time memory management (GETSP/RELSP), Record Manager label processing, and SCOPE label processing all are discussed further in the Record Manager Reference Manual.

Macro	Function
File Creation and Maintenance Macros	
FILE	Creates file information table (FIT)
FETCH	Retrieves value of any field in FIT
STORE	Sets values in fields of FIT
File Initialization and Termination Macros	
OPENM	Prepares a file for processing
CLOSEM	Terminates file processing, initiates label processing
Data Transfer Macros	
GET	Transfers data from file to working storage area
GETP	Retrieves a portion of a record from a file
PUT	Transfers data from working storage area to a file
PUTP	Transfers a portion of a record to a file
CHECK	Determines completion status of I/O operations
File Positioning Macros	
SKIP	Repositions file backward or forward
REWINDM	Rewinds volume to beginning-of-information
SEEK	Provides overlap between I/O and processing by positioning while processing
File Updating Macros	
DELETE	Deletes record from file
REPLACE	Replaces record or entry in file

Boundary Condition Macros

WTMK	Records a tape mark on a tape file
WEOR	Records end of a section
ENDFILE	Records end of a partition

A FILE control card equivalent to the FILE macro also is available.

Files created by CPC can be read or written by Record Manager once they are properly described to Record Manager. Similarly, a file created by Record Manager can be read by CPC if the file structure conforms to that required by SCOPE sequential files. A file should not be manipulated by both RM and CPC within a given run.

The reference manual for Record Manager contains details of its use. Record Manager is not further discussed in this manual.

SYSTEM COMMUNICATION MACROS

Communication between SCOPE and a program written in COMPASS is provided by the following macros. These macros exist within all of the COMPASS system text overlays CPCTEXT, IOTEXT, SYSTEXT, SCPTEXT, and TXT6RM.

SYSCOM MACRO

This macro defines standard symbols and macros.

```
SYSCOM B1
```

If B1 is present, the COMPASS pseudo instruction B1 = 1 is generated. This informs COMPASS that register B1 contains 1 throughout the program, and can affect the code produced by the R= pseudo instruction. The micro MODEL is defined as the two characters 72, 73, or 74; i.e., the CYBER 70 series model number for the installation. The symbols defined are listed below.

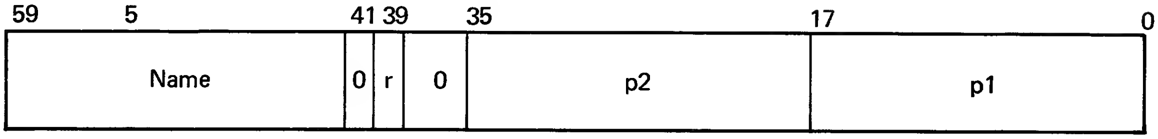
RA.SSW	=	0	Sense switches in bits 11-6.
RA.MTR	=	1	System monitor request register.
RA.ARG	=	2	Start of control statement argument list.
RA.PGN	=	64B	Bits 59-18 = program name.
RA.ACT	=	64B	Bits 17-00 = argument count.
RA.LWP	=	65B	Last word pointers for overlay load.
RA.FWP	=	66B	First word pointers for overlay load.
RA.CEJ	=	66B	Bit 59 = central exchange jump enable flag.
RA.LDR	=	67B	Loader communication word.
RA.CCD	=	70B	First word of control card image.
RA.ORG	=	100B	Origin of overlay header word for absolute programs.

SYSTEM MACRO

This macro is used for issuing system requests for which no specific system macro is provided. It is also used by many of the system action macros.

```
SYSTEM name,recall,pl,p2
```

Form in X6:



RJ SYS=

name	Three-letter name of PP program.
recall	Optional recall parameter.
pl	First parameter to PP program.
p2	Second parameter to PP program.

INTEGER MULTIPLY Opdef

This opdef provides for multiplication of 48-bit integers.

```
IXi   Xj*Xk
```

The result in registers Xi has sign extension in bits 59-48. This opdef should be deleted from the system texts when the integer multiply hardware feature is installed; COMPASS then reverts to the machine instruction of the same form, producing the octal instruction 42 ijk.

INTEGER DIVIDE Opdefs

These opdefs provide for division of 48-bit integers.

```
IXi   Xj/Xk
IXi   Xj/Xk,Bn
```

The integer quotient (fraction truncated) result in register Xi has sign extension in bits 59-48. The first form destroys register B7, and the second form destroys register Bn.

SYSTEM ACTION MACROS

The macros discussed in this section allow the user to receive status information from the SCOPE system and to change some job parameters. Calling these macros from a COMPASS central processor program results in RA+1 requests for Monitor functions or PP programs.

The macros reside in the following COMPASS system text overlays: CPCTEXT, IOTEXT, SYSTEXT, SCPTEXT and TXT6RM. All of the system action request macros call the system communication subroutine SYS=, except as noted in the individual macro descriptions; these macros do not call CPC. The subroutine SYS= resides in the library NUCLEUS.

ENDING PROGRAMS

Programs can be ended by one of two macros:

ABORT	Abnormal termination
ENDRUN	Normal termination

These functions result in a Monitor request for ABT and END, respectively; they are executed immediately by Monitor.

The ABORT function causes abrupt termination of the present program, and, if an EXIT or EXIT(S) does not appear in the control card record, will cause job termination.

```
ABORT   lfn,NODUMP
```

lfn — Not used by SCOPE 3.4; must be blank.

NODUMP — Optional parameter indicating that control card processing should resume after an EXIT(S) card. If this character string is not used, processing will resume after an EXIT card.

If neither an EXIT nor EXIT(S) card is within the control card record, the job terminates and produces the standard error dump showing the contents of the exchange package, operating registers, and memory locations near the location of the ABORT call.

The ENDRUN function is usually the last instruction to be executed in a user program. No parameters can be used with this request.

```
ENDRUN
```

Monitor will cause SCOPE to examine the job control card record and begin processing of the next control card. If the next card contains a 7/8/9 multiple punch indicating end-of-record, or is an EXIT or EXIT(S) card, the job is terminated.

FIELD LENGTH REQUEST

The field length assigned to the job may be changed with the MEMORY function. This macro also can be used to read the field length currently assigned to the job. Either central memory or ECS field length can be accessed.

```
MEMORY   type,status,recall,length,out
```

type — Field length to be accessed

CM, SCM or omitted	Central memory
ECS or LCM	Extended core storage

status — Address of reply word or address of new field length.

If length is omitted, status must contain the requested field length in bits 30-59, with bits 0-29 containing zeros

If current field length is to be returned, bits 0-59 should contain 0 originally.

recall — Optional recall parameter

length — Optional parameter giving number of words in new field length

out — Optional parameter indicating action if request cannot be satisfied:

blank	Abort job
non-blank	Continue job with present field length

The status word is also the recall reply word. Bit 0 will be set to 1 when the function is complete.

DAYFILE MESSAGES

A message is entered into the job or system dayfiles, and/or displayed on the operator console with the MESSAGE macro. The message will always appear on the operator console B display. It is printed along with the dayfiles only when specifically indicated by the macro parameters. This macro expansion calls the system communication subroutine MSG=.

The message will flash for operator attention if its first character is $. If the pause bit is set when MESSAGE is called (bit 12 of word RA+0), the message will flash and the program will halt. The program will not continue until the operator enters a command which clears the pause bit.

```
MESSAGE  addr,display,recall
```

addr — First word address of the message

display — Indicator specifying message disposition:

omitted	Display on A and B console displays and record in system and job dayfiles
LOCAL	Display on B display and record in job dayfile
other non-blank	Display on B display but do not record elsewhere

recall — Optional recall parameter; if non-blank, MSG= will construct a status word

Within the program the message must be stored in display code and must not contain any characters with display code values greater than 57 since these cannot be displayed on the console. Maximum message length of 80 characters is established by the dayfile processing routine; 40 characters will appear on each line. Messages exceeding 80 characters will be truncated. Those shorter than 80 characters must be terminated by a word with zeros in bits 0-11.

RECALL

RECALL causes the program to relinguish control of the central processor. The conditions that determine when the job will regain control of the processor depends on the form of the macro used.

Periodic recall results from:

```
RECALL
```

Control will return to the user program as soon as Monitor has accepted the request and returns control. The request is executed by Monitor itself. Once control is regained, the user must determine whether the condition that required recall is still present. This form of the RECALL macro calls the system communication subroutine RCL=.

Automatic recall results from:

```
RECALL  addr
```

addr — Address of a word (usually the first word of a FET) which will have bit 0 set to 1 before control returns to the user program.

If addr is the first word of a FET and error or end-of-file bits are set in the code and status field of the FET, control will return to a user OWNCODE routine if such routines have been established by setting the EP or UP bits and specifying OWNCODE addresses. This form of the RECALL macro calls the system communication subroutine WNB= (wait not busy).

Since recall may be entered when an input/output operation is initiated, the RECALL macro is needed only if some useful processing can be done in the time the input/output operation is being completed.

STATUS INFORMATION

TIME AND DATE FUNCTIONS

The user can determine the date and time in several formats by accessing clocks kept internally by the system.

CLOCK	Current system clock in hours, minutes, and seconds
DATE	Current date established at deadstart time when the system was loaded
JDATE	Current date in format yyddd for year and date
RTIME	Real time clock maintained by Monitor, in fractional seconds
TIME	Central processor time allowed and used by job

Each of these functions requires the user to identify a status word. The system will return the requested information before clearing location RA+1 to mark the function complete.

Each of the above functions calls the system communication routine SYS=.

The macros, and the format of the status returned, are given below.

The system clock is that established when the operator loads the system. If the operator enters a time equivalent to the outside clock, it is shown in the format hours, minutes, and seconds. If the time has not been entered, elapsed time since deadstart is shown and marked by a leading asterisk in the status word.

CLOCK status

```
59                      35                    17                0
+-----------------------------------------------------------------+
|         h      h     .      m      m     .      s     s     .  |
+-----------------------------------------------------------------+
```

The date returned is that typed by the operator when the system was loaded. Its format is display code, and generally is mm/dd/yy for month, day, and year; this order may be changed at installation option. A leading and trailing blank appear.

```
DATE status
```

59	35	17	0
m m /	d d /	y y	

Date in a format suitable for calculating elapsed days is returned with JDATE. Five display code characters appear in the low order position; the first two digits are the year, the last three the number of the day in the year.

```
JDATE status
```

59	29 0
Zeros	y y d d d

The real time clock is that maintained by Monitor for purposes such as determining peripheral processor time used. The status word will show seconds in bits 12-35 and units of 4096ths of a second (244 9/64 microseconds) in bits 0-11.

```
RTIME status
```

59	47	35 0
PP Queue Entry Count	undefined	Seconds Times 4096

The job time limit is that requested on the job card or assigned by installation default. Central processor time used is shown in seconds and milliseconds.

```
TIME status
```

59	35	11 0
Time Limit (Seconds)	CP Time (Seconds)	Milliseconds

STATUS FUNCTION

The STATUS function provides a user program with information about system resources. Two types of information are available depending on the value of the x parameter as described below.

The call to this macro is:

```
STATUS   list,x,recall
```

list — Address of a header word containing the length of the area in which status information is to be returned. The status area begins at list + 1

x — x = 1 will map available space on all public rotating mass storage devices.
x = 2 will return system information concerning files assigned to the user program.
x = 3-777 reserved by SCOPE; 1000 to 7777 reserved for installation use.

recall — Optional recall parameter

Format of the header word must be:

59	47	35	23	11	0
Zeros	List Length	Length Return	(Reserved)	Zeros	a

list length — Number of words, excluding this header word, to be used for return information; must be set by user to other than 0

length return — Number of status words returned; set by SCOPE when list is complete

a — a must be set to 0 before issuing a STATUS call

The header word is also the auto recall reply word: when bit a becomes 1, the request is complete.

When x = 1, the system returns one word of information for each public rotating mass storage device available with the default allocation flag set in the RBR. Format is:

59		47	35	23	17	11 0
0	Status	Device Type	EST Ordinal	Chan	Eq	Available PRU's

status 9 bit binary field:

000	Available, off, not in use
xnN	Assigned to control point nN (4 bits); N is octal digit
020	Files on device are in use
040	Unloaded pack
120	Contains SCOPE system routines
220	Contains permanent file directory
320	Contains SCOPE system and permanent file directory
4xx	Permanent file device
5xx	Contains SCOPE system and permanent files
6xx	Contains permanent files and permanent file directory
7xx	Contains permanent files, permanent file directory, and SCOPE system routines

device type Hardware mnemonic in display code:

AA	6603 disk
AB	6638 disk
AC	6603-II disk
AD	865 drum
AF	814 disk file
AL	821 multiple disk drive
AM	841 disk file
AP	854 disk drive

EST ordinal Position of entry for device in equipment status table (12-bit binary field)

chan Channel number by which device can be accessed

eq Equipment (controller) number to which device is connected

PRU's Number of PRU's, divided by 100 octal, of space remaining on the device; value of 7777 indicates at least 262,100 PRU's available

When x = 2, a list of file names (left justified, zero filled) is placed in every third word of the list area by the user program. If the file exists, the file name will be replaced by the first three words of the file name table (FNT). If the file does not exist, the file name will be zeroed out. Information in the FNT is used by some compilers. The FNT is further explained in the System Programmer's Reference Manual.

DEPENDENT JOB COUNT

The dependency count of a job within a dependent string can be decremented from within a user program. This count can also be decremented by a control card. The concept of dependent jobs is explained in section 4 with the TRANSF discussion. Jobs in a dependent string will not execute until their dependency count is zero.

To decrement the count of a job dependent on the currently executing job, the macro is:

```
TRANSF  list
```

list — Address of the beginning of a list containing the names of the jobs to have the dependency count reduced.

Names in the list should be left justified with zero fill. The last word must be all zeros.

The TRANSF macro expansion calls the TRANSR subroutine.

READING CONTROL CARDS

With the CONTRLC function a central processor program can read or backspace within the control card record for the job. When the function is executed, the pointer to the next control card is moved. The user is responsible for the resulting position of the control card pointer.

```
CONTRLC  status,function,dfile,crack
```

status — Address of a status reply word; or, if function is blank, address of the function request word

function — Control card pointer repositioning:

READ	Read next control card and place in locations RA + 70 through RA + 77 with zero fill if necessary
BKSP	Backspace control card pointer; for the first BKSP issued, this will cause the pointer to identify the currently executing card

If function is blank, word status is assumed to contain an octal code in bits 0-17 specifying read or backspace:

000010	Read
000040	Backspace

dfile Optional dayfile indicator. If non-blank, the card image is to be sent to the dayfile

crack Optional parameter; if non-blank, parameters from the card are to be placed in locations RA+2 through RA+53, aligned as shown below

The status word is the recall reply word. Bit 0 is set to 1 when the function is complete. If a read is requested but no more control cards exist, bit 4 of the status word is set to 1 to indicate an end-of-record. Similarly, if a backspace is issued but no previous control cards exist, bit 4 is set to 1.

If parameters are returned to locations RA+2 through RA+53, the format of these locations will be the same as that resulting from the initial read of a control card. Keywords and parameters are left justified binary zero filled in bits 18-59. A code specifying a separator, terminator, or continuation character appears in the lower four bits.

00	continuation	06	minus
01	comma	07	blank
02	equals	10	semi-colon
03	slash	16	other
04	left parentheses	17	terminator
05	plus		

A full word of zeros indicates the last parameter if the code in the lower 4 bits of the previous word is 17. Otherwise, additional control information follows.

PROGRAM RECOVERY

RECOVR FUNCTION

The RECOVR macro allows a user program to gain control at the time that normal or abnormal job termination procedures would otherwise occur. Initialization of RECOVR at the beginning of a program establishes the conditions under which control is to be regained and specifies the address of user recovery code. If the stated condition occurs during program execution, control returns to the user code. RECOVR macro expansion calls the SETUP. subroutine.

RECOVR is concerned with conditions that affect job execution. The conditions under which SCOPE will return control to the user, and the octal values that will select them in the call to RECOVR, are:

Arithmetic mode error	001
PP call or auto-recall error	002
Time or storage limit exceeded	004
Operator drop, kill, or rerun	010
System abort	020
CP abort	040
Normal termination	100

Conditions can be combined as desired, with octal values up to 177 allowed in the flag field of the call to RECOVR.

At least five seconds of central processor time always will be available for user code execution. RECOVR makes the exchange jump package and RA+1 contents available to the program if user recovery code is executed, and gives the user the option of having normal or abnormal job termination output.

Initialization of RECOVR within code at the beginning of a program results in an entry in a stack of requests for PP program RPV. Although RPV can be called directly by a Monitor request in RA+1, use of the RECOVR utility is preferable for all except stand alone system utilities because SCOPE routines themselves use this capability. Only one set of recovery conditions can exist within RPV, but RECOVR allows up to five user and system sets of flags and code for each program. The last RECOVR initialization will receive control first.

A checksum of the user recovery code can be requested during initialization. If flagged conditions subsequently occur, RECOVR will again checksum the code before returning control to it. This gives some assurance of user code integrity before it is executed.

RECOVR is initialized from a COMPASS program with:

```
RECOVR  name,flags,checksum
```

name — Address of code to be executed if flagged conditions occur; a return jump will be made to this location

flags — Octal value for conditions under which recovery code is to be executed, as outlined above; default is 77

checksum — Last word address of recovery code to be checksummed; 0 if no checksum

If one of the flagged conditions occurs, the address of the exchange jump package will be in register B1 and the RA address in B3. Register A1 will contain the address of the list of the parameters passed in B1-B3. Register B2 will contain a 0; if the recovery code sets B2 to a non-zero value, or if the code contains an ENDRUN macro or an RA+1 request for END, normal job termination procedures will follow. Otherwise, abnormal job termination procedures will follow recovery code execution.

If a program calling RECOVR contains overlays, both the call to RECOVR and the user recovery code should be a part of the level 0,0 code.

The exchange jump package returned by RECOVR is in the format shown with the Dump discussion, with the system error code that caused recovery code execution in bits 0-17 of the first word. If the P register shows zero in the package because a mode error occurred, bits 31-47 of RA + 0 (figure 11-2) will contain the P register value. System error codes that may be returned are:

Condition	Error Code	RECOVR Mask
Normal termination	0	100
Requested time limit exceeded	1	004
Arithmetic mode error	2	001
Illegal parameter passed to PP	3	020
CP program requested abort	4	040
PP program requested abort	5	002
Operator dropped job	6	010
Operator KILL	7	010
Operator initiated job rerun	8	010
CP abort (ABT + bit 36)	9	040
ECS parity error	10	020
Required auto-recall status missing	13	002
Job hung in auto-recall	14	002
Requested mass storage limit exceeded	15	004
xxx not in program library	16	002

Both the FORTRAN and FORTRAN Extended languages contain RECOVR subroutines as detailed in their respective manuals.

CALLING RPV DIRECTLY

The PP program RPV can be called by setting RA+1 as follows:

59	41		39	35	23	0
RPV		1		Flags	Recovery Address	

The code at the recovery address should allow for a 21 octal word array to be returned. Control will be returned to word 22.

An optional checksum of the recovery area can be requested in the user call. If the word at the recovery address contains all zeros, no checksum will be taken. If the upper 30 bits contain the last word address of the recovery area, a checksum of the code address+21 through last word address will be made and stored at the recovery address+1.

FILE ACTION MACROS

Each of the following functions addresses a file by its logical file name. A file environment table must exist for the file before its residence and use can be specified. The FET creating macros may be used, or the programmer can construct his own FET conforming to the format expected by the system.

When any file action request is issued, values are returned to the device type, disposition code, and FNT pointer fields in the FET.

All these functions, with the exception of READIN and WRITOUT, expand to a sequence of code that includes a return jump to routine CPC. READIN and WRITOUT bypass CPC by calling the random indexed record processors directly. For the other functions, CPC will call the appropriate PP routines to carry out the function specified.

Files manipulated by the following functions should not be manipulated by the functions described in the reference manual for the Record Manager within the same run.

The macros which call CPC are contained in the CPCTEXT system text overlay.

FILE REQUEST

File residence can be specified by a REQUEST control card or macro, with the same results. Section 4 presents further information about the use of the parameters in this macro.

File action requests must reference the logical file name (lfn) of the file. If the file is a member of a multi-file set, all functions must reference the lfn of the set member. No function except REQUEST may be issued using the set name.

REQUEST FUNCTION

The REQUEST function informs the system of file characteristics.

```
REQUEST addr
```

addr is the first word of a variable length parameter list constructed by the user. The list must be at least two words long; maximum length required is that which supplies the parameters indicated by bits set in the flag field.

The parameter list must have the form:

59	47	35	23	17	11 0	
Logical File Name				0 (Status Return)		1
Flags	EST Ordinal	Flags			Device Type; Allocation	2
Volume Serial Number			(CDC reserved)			3
Family Pack Name				P or K	ECS Buffer Size	4
Magnetic Tape File Header Label Information						5
						6
						7
						8
Magnetic Tape File Header Label Information						9

Parameters have the following meaning:

lfn — Logical file name, left justified, zero-filled.

status return — Initially, user should set to zero. The system returns the codes given below.

flag fields — Each bit is a flag for a particular condition listed below.

EST ordinal — Equipment status table ordinal of a specific device; as with the eq parameter of the REQUEST card, it should not be used except under controlled circumstances.

device type and allocation

Bits 6-11 are the octal device type code listed in the FET discussion of section 11; allocation styles of that device (except tape units) are installation defined.

volume serial number

Volume serial number identifying a magnetic tape for automatic assignment. Bit 25 of word 2 (addr+1) must be set if a VSN is given. When given, VSN must be right justified with display code zero fill. Binary zeros in this field indicate a scratch tape.

family pack name

1-7 characters of family pack name left justified, zero-filled. Bit 16 of word 2 (addr + 1) must be set if this parameter is given.

ECS buffer

If the file is to be buffered through ECS, bit 33 of word 2 (addr + 1) must be set. The size of the buffer must be in bits 0-11, with bits 12-17 showing a display code P if the size is in pages, or showing a K if the size is in thousands of words.

tape label fields

Label information for normal or extended label processing, formatted as shown below. Normal label processing is assumed unless bit 49 of word 2 (addr + 1) is set.

The flags are individual bits that should be set to 1 to indicate the following conditions. Otherwise the bits should be 0.

Bit Number	REQUEST Card Equivalent	Meaning
51	MN	7-track or 9-track tape may be assigned
49	none	Parameter list words 5-9 have extended label processing format
48	none	Parameter list words 5-9 have SCOPE 3.4 label format
33	EC	ECS buffering with parameters set in word 4 (Sequential pack files cannot be ECS buffered.)
32	OV	Overflow allowed to different device if that specified in word 2 is not available; if EP bit is set, a device capacity exceeded status will be returned if no mass storage is available; permanent files will overflow only to another permanent file device
31	PF	File must reside on permanent file device
30	US	9-track tape conversion to ASCII codes
29	EB	9-track tape conversion to EBCDIC codes

28	* prefix to device type	Device to be assigned by system rather than operator
27	none	Format of operator flashing message; if set, contents of RA+70 through RA+77 are displayed; if 0, REQ constructs the message from the REQUEST parameter list
26	2 prefix to device type	Two magnetic tapes or sequential pack units requested
25	VSN	Word 3 contains a volume serial number for a magnetic tape
24	E	Magnetic tape is labeled currently
23	NS	Non-standard labels on tape should be considered data, not labels, by operating system
22	NR	Normal system tape read parity error processing is to be inhibited
21	Z	Magnetic tape has Z format label of SCOPE 3.3, with character 12 of VOL1 label establishing data density
20	none	Special return of error code to user; do not issue dayfile message or consult operator
19	–	Not used
18	MF	Request is for a multi-file set
17	DP	Request is for a sequential pack (If EC parameter also is present, it will be ignored.)
16	PK	Request is for a family pack
15	absence of explicit density	Magnetic tape is to be written at system default density
14	SV	Output tape to be saved
13	IU	Inhibit physical unload of tape
12	CK	Checkpoint tape request

Format of the tape label fields depends on whether normal label processing is requested. The label fields must be in display code format, with acceptable values for each field, as detailed in appendix E.

Label information for normal processing:

<table>
<tr><th>59</th><th>47</th><th>29</th><th>23</th><th>17</th><th>11</th><th>0</th></tr>
<tr><td colspan="7">File Label Name</td></tr>
<tr><td colspan="4">File Label Name</td><td colspan="3">Position Number</td></tr>
<tr><td>Edition Number</td><td>Retention Cycle</td><td colspan="5">Creation Date (yyddd)</td></tr>
<tr><td colspan="3">Multi-file Set Name</td><td colspan="4">Reel Number</td></tr>
</table>

Label information for extended label processing:

<table>
<tr><th>59</th><th>53</th><th>41</th><th>35</th><th>29</th><th>17</th><th>11</th><th>5</th><th>0</th></tr>
<tr><td colspan="3">HDR1</td><td colspan="6">File Label Name</td></tr>
<tr><td colspan="9">File Label Name</td></tr>
<tr><td>a</td><td colspan="4">Multi-file Set Name</td><td colspan="4">Reel Number</td></tr>
<tr><td>b</td><td colspan="3">Position Number</td><td colspan="3">Generation Number</td><td colspan="2">c</td></tr>
<tr><td>c</td><td colspan="4">Creation Date (yyddd)</td><td></td><td colspan="3"></td></tr>
</table>

a File label name continued

b Reel number continued

c Edition number

Once the REQUEST function is completed, bit zero of the first word of the parameter list is set to 1. In addition, bits 9-13 of word 1 may show one of the octal codes below. If so, the REQUEST function has been ignored and control returned to the program.

22	Recall bit was not set in call to PP routine REQ
24	File name table is full
26	Either requested equipment does not exist in the configuration, or all equipment of this type is already assigned to this job
30	File is already assigned to a device; the device type code will be returned to the device type field of the parameter list

Two dayfile messages result from a successful REQUEST function. The first, directed only to the operator, contains parameters corresponding to those used in the internal parameter list. After assignment, a second message is written to the job and system dayfiles reflecting the assignment. For example, if a REQUEST function is made with dt set to zero, the operator display will show no device type. If the operator assigns a 7-track tape, however, the mnemonic MT will appear in the job dayfile message.

Conflicts between dt and EST ordinal requested, or between dt requested and dt assigned by the operator, must be resolved by the operator by use of the n.YES or n.NO type-in.

OPEN AND CLOSE REQUESTS

Two functions are available for opening files:

OPEN is applicable to all files

POSMF is applicable only to labeled multi-file tapes

Files can be closed with the following functions:

CLOSE is applicable to all files

CLOSER is applicable to sequential files on tape or sequential disk packs; it gives the user control over end-of-volume processing

OPEN FUNCTION

An OPEN function is a file initialization and status checking operation. The user must issue an OPEN if:

Random files are to be processed by the user or system

User label processing is to follow

Sequential files are to be rewound without a REWIND function being issued

Otherwise, OPEN is not necessary. If an OPEN function is to be issued, it should be the first function issued on a given file; otherwise the effect of the OPEN function is undefined.

```
   OPEN  lfn,x,recall
```

The x parameter may be any of the ten values:

absent	NR
READ	READNR
REEL	REELNR
ALTER	ALTERNR
WRITE	WRITENR

The WRITE or WRITENR values of x may be used to ensure that the file circular buffer will be emptied if the job terminates abnormally before buffer contents have been transferred to an output device. The first data function following these OPEN functions must not then be a read or a forward motion function.

If the value of x is READ, REEL, ALTER, WRITE, or absent, sequential files will be rewound. Any other value of x will not reposition the file.

When an OPEN is issued, the following events occur:

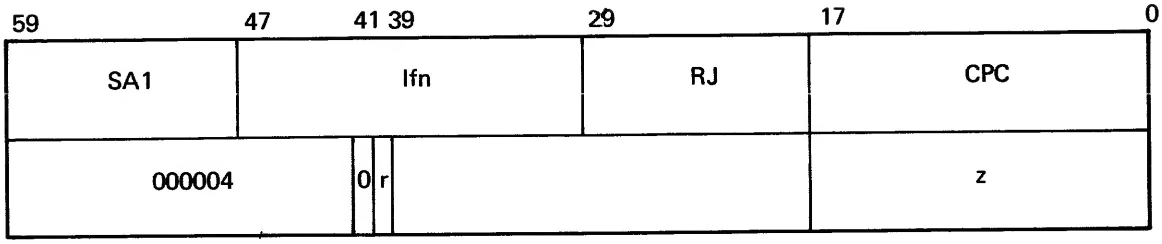
For sequential files, file position will be changed to beginning-of-information unless a no rewind is specified by using an x parameter ending in NR. The r bit in the FET is set to zero.

For labeled magnetic tape files, processing depends on the presence or absence of the XL bit in the FET. If the XL bit is set, all labels are written from or delivered to the file label buffer. If the XL bit is off and labels are being written, the HDR1 label is formatted from data in the FET label fields. If labels are being read (XL off), the HDR1 label is returned to the FET label fields and is written into the file circular buffer starting at IN provided sufficient space exits between IN and LIMIT.

For random files, if the r bit is set, any existing index is read into the index buffer. If the index record is shorter than the buffer, unused buffer space is set to zeros. If the r bit is not set, an existing index will not be read.

For all files, the physical record unit and record block sizes are returned to FET fields.

The macro OPEN generates the following code:



The z field depends on x.

160 if x is absent or ALTER
140 if x is READ
340 if x is REEL
144 if x is WRITE

120 if x is NR or ALTERNR
100 if x is READNR
300 if x is REELNR
104 if x is WRITENR

There is no difference in the action taken by the system for codes 160, 140 and 340; or for codes 120, 100 and 300.

POSMF FUNCTION

The POSMF function positions standard labeled multi-file sets. The named multi-file set is positioned to a particular file and an OPEN with rewind function is performed. The file to be opened is determined by the content of the label fields of the FET or, if the XL bit is on, the content of the HDR1 label in the label buffer. If a position number is specified in the label field or label buffer, that number specifies the file to be opened. If no position number is present, the file opened will be one which matches the fln field of the label information provided.

```
POSMF  mfn,recall
```

mfn Multi-file set name

recall Non-zero value if recall desired; otherwise blank

The position number is specified in either word 11 of the FET or the extended label buffer, depending on the label processing to be performed. If normal label processing is to occur, bits 0-17 of word 11 of the multi-file name FET may contain the position number (position numbers begin with 1 for the first file). For extended label processing to occur, the XL bit must be set (bit 41 at mfn+1). The position number is expected to be in the ANSI standard position field of a record formatted as an HDR1 label within the label buffer. A fatal error exists if HDR1 is not found within the label buffer.

If the position number is 0 and no fln is provided, the set will be positioned at the beginning of the next file. OPEN procedures for an existing file will follow. If the position number is 999 in the FET or 9999 in the label buffer, the set will be positioned after the last member file and OPEN procedures instituted for a new file.

End-of-set status (21 in bits 9-13 of mfn) is returned to the FET for the multi-file set if the explicit or implied position number is greater than the last member of the set. The position field in the FET will be one greater than that of the last member file.

The POSMF macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th></th><th>39</th><th>29</th><th>17 0</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>000110</td></tr>
</table>

CLOSE FUNCTION

A CLOSE function is a file terminating operation. The user must issue a CLOSE if:

Random files have been created or modified and a valid index is to be saved

End-of-job procedures listed below are to be initiated for a file before the actual end-of-job

Otherwise, a CLOSE is not necessary.

```
CLOSE  lfn,x,recall
```

The x parameter may be:

absent
NR
UNLOAD
RETURN

If the value of x is absent, UNLOAD, or RETURN, the file will be rewound. NR specifies that the file is not to be rewound. Both of these positionings are possible only with sequential files; positioning is not defined on files for which an index is written.

When a CLOSE is issued, the following events occur:

For sequential files, position will be changed according to the rewind associated with the x parameter.

For labeled magnetic tape files, action depends on the x parameter. If no rewind is specified and the file is positioned after a newly written record, a file mark and an EOF trailer label set will be written, then the file will be positioned immediately before the file mark. If the file is to be rewound and it is positioned after a newly written record, an EOF trailer label set will be written before the rewind is initiated.

For unlabeled tape files, four tape marks are written instead of an EOF set. Otherwise, processing is the same as for labeled tape files.

For random files, the index will be written as the last SCOPE logical record if the FET r bit is set, an index buffer is specified, and file contents have been altered since the last OPEN function was issued.

The user must empty the file circular buffer when files are being written; CLOSE will not empty the buffer.

When CLOSE/RETURN or CLOSE/UNLOAD is issued, end-of-job processing procedures occur for the named file.

Permanent files are detached from the job.

Family pack files are locked, making them unavailable for the remainder of the job.

For sequential packs, a CLOSE with rewind causes a rewind to the beginning of the current pack, which may be the first pack for the file. If an input/output request follows the CLOSE and the first pack in the file is not presently mounted, the system will instruct the operator to mount the correct pack.

For magnetic tape files, a CLOSE/RETURN decreases the number of tape units logically required by the job as indicated with the MT or NT parameter on the job card. A CLOSE/UNLOAD does not decrease this value. A CLOSE/UNLOAD or CLOSE/RETURN function issued on a member of a multi-file set acts as a CLOSE/REWIND on that member.

The CLOSE macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th>39</th><th></th><th>29</th><th>17 0</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000007</td><td>0</td><td>r</td><td colspan="2"></td><td>z</td></tr>
</table>

The z field depends on x.

150 if x is absent
130 if x is NR
170 if x is UNLOAD
174 if x is RETURN

CLOSER FUNCTION

Under SCOPE, processing of both magnetic tape and sequential pack files continues across reel or pack boundaries when data is skipped in a forward direction, read, or written. With the UP bit of the FET the user can request notification when a boundary is about to be crossed. This allows reels or packs to be processed in other than ascending order.

The CLOSER function affords a degree of user control over processing at end-of-reel or end-of-pack, depending on the setting of the UP bit when CLOSER is issued. Processing also can be terminated prematurely.

```
CLOSER lfn,x,recall
```

The x parameter is effective only if the UP bit is set when the function is issued. The x parameter may be:

absent	Rewind
NR	No rewind
UNLOAD	Rewind and unload

For magnetic tapes, the system will initiate reel swapping if the UP bit is 0 when CLOSER is issued. The file will be positioned on the next reel and file operations can continue normally. An OPEN function is not required for the second reel, but may be issued if the program is to receive the header label contents.

A reel swap is performed by the following steps:

1. If the tape is positioned after a newly written record, a volume trailer label will be written.

2. The tape is unloaded and the operator is notified that processing on that reel is completed.

3. If two units were assigned to the file, unit numbers will be interchanged so processing continues without changing tables referencing the unit.

4. The reel number of a labeled file will be incremented by one in the system label table and, if declared, in the user's FET label fields.

5. The FET completion bit is set. End-of-reel status is not returned.

If the UP bit is set to 1 when the CLOSER is issued for a tape file, the user may specify the next reel to be processed. SCOPE performs the following:

1. If the tape is positioned after a newly written record, a volume trailer label is written.

2. The tape is rewound or rewound/unloaded according to the CLOSER parameter.

3. The operator is notified that processing on that reel is completed.

4. If two units were assigned to the file, unit numbers will be interchanged.

5. The end-of-reel status and completion bits are set.

To establish the next reel to be processed, the user must enter the reel number in the FET label field (bits 0-24 in word 13) before another function is issued to the file. A following OPEN function is not required unless the program will use the header label of the new reel.

For sequential disk packs, end-of-pack status is returned by setting bit 10 of the FET code and status field when the UP bit is set. SCOPE then performs the following:

1. The processed pack unit is turned off and the operator is notified that processing is completed and the pack is to be removed.

2. If two units were assigned to the file, the unit numbers are interchanged.

3. The end-of-pack status bit is cleared if it had been set; the FET completion bit is set.

Processing wiii continue on the next pack as soon as the user issues another input/output function for the file.

The macro for CLOSER generates the following code:

<table>
<tr><td>59</td><td>47</td><td colspan="2">41</td><td>39</td><td>29</td><td>17</td><td>0</td></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td colspan="2">CPC</td></tr>
<tr><td colspan="2">000007</td><td>0</td><td>r</td><td colspan="2"></td><td colspan="2">z</td></tr>
</table>

The z field depends on x.

350 is x is absent
330 if x is NR, although result is the same as 350
370 if x is UNLOAD

READ REQUESTS

Six read functions are available for bringing information into central memory. The functions, and the main distinctions among them, are:

READ — Applicable to all mass storage and tape files. Reading stops when end-of-record is encountered.

READNS — Applicable to mass storage files only. Read does not necessarily stop at end-of-record.

READSKP — Similar to READ, but positions file to beginning of next record when the circular buffer is filled.

RPHR — Applicable to magnetic tapes in SCOPE format only. Reads the next physical record, delivering coded data in internal BCD codes (7-track). For 9-track SCOPE tapes, the data is read in packed mode and delivered with no conversion.

READN — Applicable to magnetic tapes in S and L data format only.

READIN — Applicable to all mass storage and tape files.

All of these functions read information into the file circular buffer, with the amount of information read dependent on the specific function and the size of the buffer. As information is read into the buffer, the SCOPE routines change the value of the IN pointer. This value, minus 1, is the address of the last word read. The user is responsible for using the IN pointer while removing information from the buffer, and for setting the OUT pointer to reflect the move, except when the READIN macro is called. READIN, like WRITOUT, relieves the user of responsibility for IN and OUT pointer manipulation. By means of a secondary buffer called a working storage area, READIN maintains circular buffer pointers.

As processing progresses, status information will be returned to the code and status field of the FET. If the user has the EP bit set, control will return to his program for OWNCODE routine execution when file action errors occur. Otherwise, the operator will be notified and given the option to drop the job.

The 18 bit code and status field will show the values listed below for the conditions that cause various read functions to terminate. Bits in the field have the purposes:

Bits 14-17	SCOPE logical record level number
Bits 9-13	File action error code
Bit 4	End-of-record indicator
Bit 3	End-of-file indicator if bit 4 is set
Bit 1	Mode indicator: 0 for coded, 1 for binary
Bit 0	Complete bit

For binary files, the low order octal digit of the code and status will be 3 instead of 1.

Condition	Code/Status Setting for Coded Files
End-of-information encountered	741031
Short or zero length SCOPE logical record of level xx is read	xx0021
Level 17 SCOPE logical record or level 16 mass storage file read with READNS	740031
Next PRU will not fit into circular buffer	000011
Unrecoverable file action error code ee	0ee011

File action error codes are listed in the Error Address field in the FET discussion of section 11.

When a read for a file is issued without recall, the IN pointer is updated as each PRU of data is moved to the buffer, allowing the user to remove data as fast as it is placed in the buffer. When the request is issued with recall, the pointer is not changed until the request is complete.

For S and L format files, the UBC field is set as a record is read.

All the following read functions, except READIN, expand to a two-word sequence of code which includes a return jump to routine CPC. The READIN function expands to call routine IO or IORANDM, which calls CPC.

Parameters appearing in the macros are:

lfn	Logical file name
recall	Optional recall parameter of any non-blank alphanumeric character

READ FUNCTION

The READ function is applicable to all types of files. READ causes information from the specified file to be placed in the circular buffer for the file in central memory.

```
   READ  lfn,recall
```

Reading begins as long as the circular buffer has room for at least one physical record unit. It continues until:

The next PRU will not fit into the circular buffer.

End-of-record or end-of-file is encountered.

End-of-information is encountered.

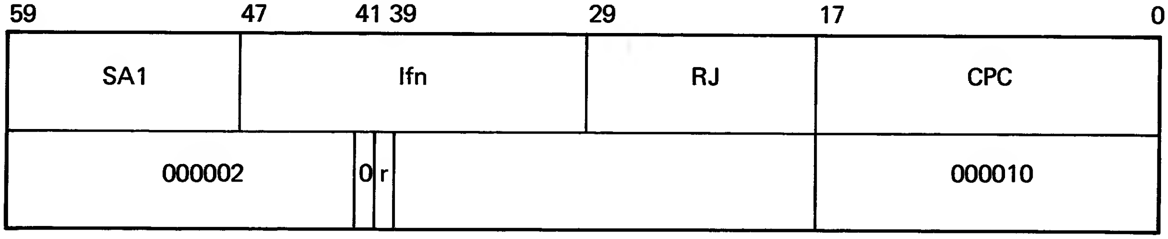
File action error occurs.

For S and L tapes, one physical record is read.

If the end-of-record bit (bit 4 of word 1 of the FET) is already set as a result of a previous READ, CPC will ignore any further READ calls as redundant until bit 4 is cleared.

For S and L tapes, the unused bit count is returned to the UBC field in the FET word 7 when the read is complete.

The READ macro generates the following code:



READNS FUNCTION

The READNS function is applicable only to mass storage files. A single READNS often results in more information being transferred to the circular buffer than a READ issued to the same file since reading does not necessarily stop at the end of a logical record.

```
   READNS  lfn,recall
```

Reading begins if the circular buffer has room for at least one physical record unit. Reading continues until:

The next PRU will not fit into the circular buffer.

Zero length SCOPE logical record of any level is read.

Level 16 or 17 SCOPE logical record of any length is read.

End-of-information is encountered.

File action error occurs.

Reading continues when end-of-record is set.

The READNS macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th></th><th>39</th><th>29</th><th>17</th><th>0</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td colspan="2">CPC</td></tr>
<tr><td colspan="2">000002</td><td>0</td><td>r</td><td colspan="2"></td><td colspan="2">000250</td></tr>
</table>

READSKP FUNCTION

The READSKP function is applicable to all types of files. READSKP is used to identify and skip records. Reading continues until an end-of-record is encountered, or the circular buffer is full. Once the buffer is full, the file is repositioned to the beginning of the next record. READSKP is halted by any conditions which halt a READ.

```
READSKP  lfn,lev,recall
```

lfn   Logical file name

lev   Optional level number 0-17. Default value is 0.

recall   Optional recall indicator.

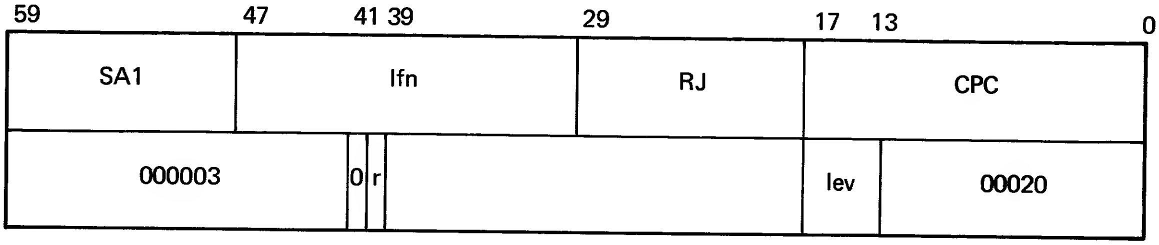
If a level parameter lev is specified, information is skipped until the occurrence of an end-of-record with a level number greater than or equal to the one specified. For S and L tapes, only a request with level 17 is recognized; any other level in the request will be ignored.

When the READSKP is executed, the end-of-record bit (bit 4 in word 1 of FET) is set, since an end-of-record is encountered in the skip to the beginning of the next record. This bit must be cleared by the user program before a subsequent READ, but not a READNS, is issued.

For S and L tapes, the user should set the MLRS field before the READSKP is issued. If this field has a 0, the system will set it to 512 words for an S tape and to LIMIT–FIRST–1 for an L tape.

An end-of-reel condition on a magnetic tape file with the UP bit set will terminate the skip of a READSKP even if the beginning of the next record has not been encountered. Otherwise, reel swapping takes place under system control.

The READSKP macro generates the following code:



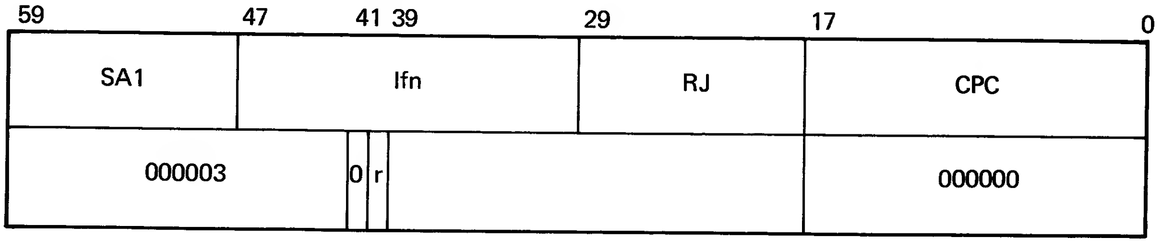
RPHR FUNCTION

The RPHR function is applicable only to magnetic tapes in SCOPE format. RPHR causes all information existing in the circular buffer to be discarded and the next physical record to be read into the buffer.

```
RPHR  lfn,recall
```

For coded 7-track files, data is converted from external to internal BCD only. Conversion to display code is not made. No conversion takes place for 9-track tapes; the data will appear as written. SCOPE tapes are always written to contain exact multiples of central memory words by filling the last word with zeros.

The RPHR macro generates the following code:



READN FUNCTION

The READN function is applicable only to magnetic tape in S or L format. READN allows maximum tape throughput; as long as the user provides space in the circular buffer for two records and their header words, tape reading continues without releasing and reloading the read routine between physical records. This gives maximum utilization of interrecord gap time.

```
   READN  lfn,recall
```

Before this function is issued, the MLRS field of the FET (bits 0-17 of word 7) must be set to the largest physical record that will be encountered. File mode must also be set.

Reading continues until:

The next record will not fit into the circular buffer.

End-of-file is encountered.

End-of-information is encountered.

File action error occurs.

The header word that precedes each physical record in the circular buffer is generated by the system; it does not exist on the tape. The format of the header word is:

59	29	23	17 0
	UBC		CM words

CM words   Number of 60-bit words in the physical record

UBC   Number of bits in the last word that are not valid data

After each complete physical record has been placed in the buffer, the system moves the IN pointer to reflect both the header and data.

The READN macro generates the following code:

59	47	41	39	29	17 0
SA1	lfn			RJ	CPC
000003		0	r		000260

READIN FUNCTION

The READIN function is applicable to all mass storage and tape files. SCOPE indexed files are limited to mass storage. READIN employs a user-provided working storage area as well as the file circular buffer. The user deals only with data in the working storage area; the system handles the circular buffer and the IN and OUT pointers of the FET.

Format of the READIN macro depends on the structure of the file being accessed. The second parameter is required only if the file is a random SCOPE indexed file with a name or number index.

When READIN is executed, data from the circular buffer is placed in the working storage area. The amount of information transferred depends on file mode:

> For binary files, READIN fills the working storage area unless an end-of-record or end-of-information is encountered before the area is full.
>
> For coded files, information is moved to the working storage area until a 12-bit zero byte (end-of-line indicator) is encountered or the working storage area is full. When a zero byte is encountered, two blanks are substituted and the remainder of the area is filled with blanks. If a zero byte is not met before the working storage area is full, the remainder of the line is skipped. The next READIN request will obtain the next line rather than the end of the first line.

READIN will issue calls to READ through CPC as needed. If the data in the buffer does not satisfy the READIN request, a READ with recall is issued. Therefore, the user does not gain control until his request is satisfied.

If a working storage area is not specified, a READIN request has no effect, except as described below for SCOPE indexed random files.

READIN makes a check of the I/O progress immediately prior to returning to the user program. A READ without recall will be issued if the circular buffer is not already busy and it is more than half empty, so that input/output is buffered with subsequent computing by the user program.

Sequential or random files are read with the following macro:

```
  READIN  lfn
```

When an end-of-record or end-of-file is encountered during a read of a sequential file, the user regains control immediately, with the X1 register showing the state of the request. Filling of the working storage area ceases. The next READIN request will begin with the next record.

Status information in the X1 register may be:

positive zero — Requested number of words was read and the function completed normally.

positive non-zero — The working storage area was not filled because the remainder of the logical record contained too few words when the READIN was issued. X1 contains the address of the first unfilled word, or if no data was transferred, the first word address. For coded files, this is always the first word address.

negative non-zero — No data was transferred to the working storage area because an end-of-file or end-of-information was encountered.

When a SCOPE indexed random file has named or numbered records, READIN positions the file to the desired record.

```
READIN  lfn,/name/

READIN  lfn,n
```

/name/	Name of record
n	Number of record

When a READIN is issued for such a SCOPE indexed random file, the current contents of the circular buffer are destroyed when the IN and OUT pointers are set equal. Then, the mass storage address corresponding to the record number or name is copied from the index into FET word 7, and a READ request with recall is passed through CPC. On return from the READ, the procedures for a READIN without a name or number parameter are followed. If a working storage area is specified in the FET, the beginning of the record is copied into it and the FET pointers are adjusted. If no working storage area is specified, no further action occurs; however, the file has been positioned and reading of the desired record has been initiated by READIN.

Any remainder of the record can be read by subsequent READIN requests that do not identify the record by name or number. After an end-of-record is encountered on a random file, further READIN requests specifying only a file name will not initiate reading of the next record, as they would on a non-random file. To start reading the next record, or some other record on a random file, a READIN with a record name or number must be issued.

When a record is located by a READIN request containing its name or number, the number of the record is stored in word 8 of the FET, making it possible to read the next record with:

```
READIN  lfn,0
```

The system interprets this statement as record n+1. Consequently, by starting a new record with a request that identifies record number zero, the list of records as given in the index can be read. However, if the calling program did not stop before overshooting the end of the index, there would be an error return from READIN on the last+1 record.

The code generated by the READIN macro depends on the second parameter. For no parameter, a name parameter, and a number parameter, respectively, the code is:

59	29	17 0
	RJ	IOREAD
		lfn

59	29	17 0
	RJ	IORR
		lfn
name		

59	29	17 0
	RJ	IORR
		lfn
		n

WRITE AND REWRITE REQUESTS

Information is transferred from the file circular buffer to a storage device when one of the write functions is issued. These functions, and the main distinctions among them, are:

WRITE Applicable to mass storage and tape files; writes at end-of-information

WRITER Applicable to mass storage and SCOPE tapes; writes a short or zero length PRU to indicate end-of-record

WRITEF Applicable to mass storage and magnetic tape files; writes an end-of-file indicator

WPHR Applicable to magnetic tapes in SCOPE standard format only; writes a single physical record; the only write function that expects coded data in internal BCD format. No conversion is performed for SCOPE 9-track coded tapes.

WRITEN Applicable to S and L data format tapes only

WRITOUT Applicable to mass storage and tape files; the only write function in which the system, rather than the user, manipulates the buffer pointers of the FET

REWRITE Applicable to mass storage only; rewrites record of same length

REWRITER Applicable to mass storage only; writes an end-of-record indicator for a rewritten record

REWRITEF Applicable to mass storage only; writes an end-of-file for a rewritten file

WRITIN Applicable to mass storage files to be rewritten only; analogous to WRITOUT using REWRITE rather than WRITE

The system sets the OUT pointer when data is removed from the buffer. The user must manipulate the IN pointer as he places information in the buffer, as explained under the Buffer Pointer Fields of the FET discussion.

When S and L tapes are being written, the MLRS and UBC fields in the FET must be set by the user to indicate the size of the record before a write is issued.

Status information and error codes are returned to the first word of the FET as the file is written. If the user has the EP bit set, control will return to his program for OWNCODE execution when file action errors occur. Otherwise, operator will be notified and given the option to drop the job.

Parameters that appear in the write macros are:

lfn	Logical file name
recall	Optional recall parameter consisting of any non-blank letter/number character string

WRITE FUNCTION

The WRITE function transfers information from the file circular buffer to the file storage device. WRITE is applicable to both mass storage files and tapes.

```
WRITE  lfn,recall
```

For mass storage files and tapes in SCOPE format, only full PRU's are written. The size of the PRU depends on the storage device. Writing continues until:

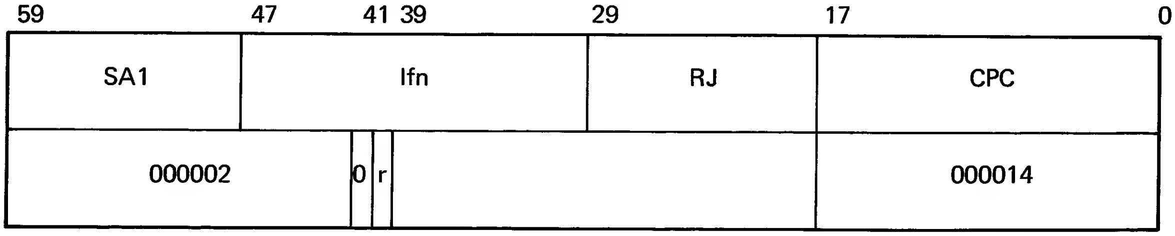
The buffer is empty.

Data in the buffer does not fill a PRU.

A following WRITER request will empty the buffer.

For tapes in S or L format, only one record is written for each request. The length of the record is determined by the value of the IN and OUT pointers. If the record length exceeds the MLRS field value (bits 0-23 of word 7) in the FET, the job terminates with an error.

The WRITE macro generates the following code:



WRITER FUNCTION

The WRITER function causes the circular buffer to be emptied and an end-of-record indicator to be written. For mass storage files and tape files in SCOPE format, a short or zero length PRU is written. For S and L format tape files, WRITER is equivalent to WRITE.

```
WRITER  lfn,lev,recall
```

lfn Logical file name

lev Optional level number 0-17. Default value is 0.

recall Optional recall indicator

WRITER is processed the same as WRITE, with the following additions:

> For mass storage files and tapes in SCOPE format, the data in the circular buffer is written out followed by an end-of-record marker. A zero length PRU is created if necessary; otherwise a short PRU will exist. If the level parameter is present, it will be included. If the buffer contains no data when WRITER is issued, a zero length PRU is created. If the specified level number is 17, the system will change the WRITER request to a WRITEF.

The WRITER macro generates the following code:

<table>
<tr><td>59</td><td>47</td><td>41</td><td></td><td>39</td><td>29</td><td>17</td><td>13 0</td></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td colspan="2">CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>lev</td><td>00024</td></tr>
</table>

WRITEF FUNCTION

The WRITEF function produces an end-of-file. Any information in the buffer is written out before the end-of-file is written.

```
WRITEF  lfn,recall
```

For mass storage files and tapes in SCOPE format, WRITEF produces a logical end-of-file mark written as a zero length PRU of level 17. Data in the buffer is written out and terminated by a zero level end-of-record before the end-of-file marker is written. If the buffer is empty and the last operation was a write, a zero length PRU is written as an end-of-record indicator before end-of-file.

For S and L tapes, data in the buffer is written to tape and followed by a physical tape mark.

The WRITEF macro generates the following code:

<table>
<tr><td>59</td><td>47</td><td>41</td><td></td><td>39</td><td>29</td><td>17 0</td></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>000034</td></tr>
</table>

WPHR FUNCTION

The WPHR function is applicable only to magnetic tape in SCOPE format. It causes all information in the circular buffer, to a limit of 512 words, to be written as a single physical record. Data to be written to 7-track tape must be in internal BCD codes. Only internal to external BCD conversion is performed before writing; no conversion is performed for 9-track SCOPE tapes.

```
WPHR  lfn,recall
```

If the buffer contains fewer than 512 (decimal) words, the IN and OUT pointers in the FET are set equal when writing is completed to show an empty buffer. If the buffer contains more than 512 words, only the first 512 words are written. The IN and OUT pointers will be set by the system to show that more data exists in the buffer. Status returned is 10, indicating device capacity exceeded.

A WPHR issued for any device other than magnetic tape in SCOPE format is ignored. A 22 status is returned to show an illegal function call, terminating the job.

The WPHR macro generates the following code:

<table>
<tr><td>59</td><td>47</td><td>41</td><td></td><td>39</td><td>29</td><td>17 0</td></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>000004</td></tr>
</table>

WRITEN FUNCTION

The WRITEN function is applicable to magnetic tape in S or L format only. It allows maximum use of the interrecord gap time as long as the user provides at least two records and their control words in the circular buffer.

```
WRITEN  lfn,recall
```

Writing continues until:

Buffer is empty.

End-of-reel is encountered.

File action error occurs.

No action takes place if the buffer is empty.

The user must provide a header word immediately preceding each record in the buffer. This header is not physically written on the tape. Its format is:

59	29	23	17 0
	UBC		CM Words

CM words Number of 60-bit words in the physical record

UBC Number of bits that are not valid data in the last word

The system compares the MLRS and UBC fields in the FET using information from this header.

The OUT pointer is not changed to reflect the move until after each complete record has been written to tape. The user should not move the IN pointer beyond the header word until the header and the complete record are in place, or an error will result.

The WRITEN macro generates the following code:

59	47	41		39	29	17 0
SA1	lfn				RJ	CPC
000002		0	r			000264

WRITOUT FUNCTION

The WRITOUT function is applicable to all mass storage and tape files, with SCOPE indexed files being limited to mass storage. It employs a user-provided working storage area as well as the file circular buffer; when the buffer is full the system issues a WRITE function to transfer data from the buffer to the file storage device. With SCOPE indexed files, the user has the option of using either WRITOUT to position a file and manage the circular buffer himself, or providing a working storage area and letting the system manage the buffer. Otherwise, the user deals only with data in the working storage area; the system handles the circular buffer and both the IN and OUT pointers of the FET.

When WRITOUT is executed, data in the working storage area is transferred to the circular buffer. No record boundaries will be assumed, with all data placed in the buffer by WRITOUT being considered a single logical record. Until the user issues a WRITER or WRITEF to empty the buffer and write an end-of-record or end-of-file mark, a single record exists. The system will empty the buffer as necessary to accommodate new data being moved into the buffer. As with the READIN function, the system buffers input/output with computing by checking the buffer just before returning to the calling program, and issuing a WRITE without recall if the buffer is more than half full. WRITE functions with recall are issued when it is necessary to empty the buffer before carrying out the WRITOUT request, so that the WRITOUT function will be complete before control returns to the user program.

The amount of data transferred from the working storage area to the buffer depends on the file mode:

For binary mode files, the entire working storage area is transferred.

For coded mode files, trailing blanks are removed and a 12-bit zero byte is inserted to indicate end-of-line.

Sequential and random files without indexes are written with:

```
WRITOUT  lfn
```

A WRITER function must be used to terminate a record. If a working storage area does not exist for a sequential or random file, the WRITOUT is ignored with no error indication.

An additional parameter is required when the file has SCOPE indexed records.

To declare the beginning of a SCOPE indexed record, one of these forms of the macro is used:

```
WRITOUT  lfn,/name/

WRITOUT  lfn,n
```

/name/ Name of record

n Number of record; if n is 0, the number will be one greater than the last number, with the first record being numbered 1

To continue writing the same record, this form is used:

```
WRITOUT  lfn
```

To terminate the record, the WRITER macro should be used, although the system will issue WRITER under circumstances noted below.

```
WRITER  lfn
```

An alternate method of processing SCOPE indexed records is to use WRITOUT with a record identifier, then fill the circular buffer directly and issue a WRITE request without using the working storage area. A WRITER request is still needed to terminate the record.

When a WRITOUT identifying an indexed record is issued, the system performs the following:

> If the buffer contains data from a previous WRITOUT, or the last operation was a completed write rather than write end-of-record, a WRITER occurs.
>
> The IN and OUT pointers are set equal to indicate an empty buffer and the FET status is set to show that write was completed.
>
> The random file index and the eighth word of the FET are set to the correct record.
>
> The working storage area is transferred to the circular buffer as the beginning of the new record identified in the WRITOUT.
>
> If the buffer contains at least one PRU of data, WRITE is called.

When a working storage area does not exist for an indexed file, or the length of the area is 0, the same procedures occur with the omission of any transfer of data to the buffer.

The code generated by the WRITOUT macro depends on the parameter list. For no second parameter, a name parameter, or number parameter, respectively, the code is:

59	29	17 0
	RJ	IOWRITE
		lfn

59	29	17 0
	RJ	IORW
		lfn
name		

59	29	17 0
	RJ	IORW
		lfn
		n

REWRITE FUNCTIONS

The functions REWRITE, REWRITER, and REWRITEF update records in existing mass storage files. A fourth rewrite function, WRITIN, can be used similarly to WRITOUT; it can be used in conjunction with REWRITE, as the WRITOUT function with WRITE, and the REWRITER function should be used to terminate the record rewritten. These functions do not change the total amount of mass storage assigned to the file, nor do they update any index which may be associated with the file.

All of these functions call for writing in place, not writing at end-of-information. Since the system cannot determine the length of the original record, it offers no protection from over-writing or under-writing and does not issue diagnostics when these conditions occur. The system guarantees only that a rewritten record does not extend beyond the file end-of-information, with writing taking place up to that point and a diagnostic issued if the program attempts to go beyond that point. End-of-information is never moved. The index record existing at the end of random file is not protected.

Rewrite functions are similar to WRITE, WRITER, and WRITEF. Parameters for the macros are the same.

```
REWRITE   lfn,recall

REWRITER  lfn,lev,recall

REWRITEF  lfn,recall
```

The user is responsible for knowing file structure before and after the rewrite. A minimum of one PRU is transferred from the circular buffer to the file each time a rewrite function is issued. Writing always begins at the current file position. Therefore the user must see that the file is positioned properly before writing takes place.

The amount of information rewritten for each call depends on the amount of information in the circular buffer, with the minimum amount being one PRU which may include a short or zero length PRU. When a SCOPE logical record is to be replaced with a record of the same length in a single rewrite operation, REWRITER should be used. A longer record may require REWRITE and REWRITER, depending on the buffer size.

When the new record is not the size of the original record, the resulting file may have spurious records. Short replacement records, where the original record was contained in a single PRU, or the replacement record extends into the last PRU of the original record, do not cause difficulties. When the new record occupies fewer PRU's than the original, however, the end of the original record will remain in the file. As an example, consider an original 120-word record occupying a full PRU of 64 words and extending 56 words into a second PRU. Replacing the record with 60 words produces a short PRU in place of 64 words of original data. The 56 words of the second PRU of the original record remain in the file, since mass storage allocation never is changed by a rewrite.

A similar condition is created when the replacement record extends beyond the PRU's of the original record. Since the beginning of the next record in the file will be overwritten, its usefulness is destroyed, but the remainder of the record will still reside in the file.

When REWRITEF is issued, a zero length PRU containing a level 17 end-of-file indicator is written. If issued when the file is positioned at any point other than the original end-of-file, two end-of-file indicators will exist on the file.

When random files are being rewritten, the methods of writing and the results of under-writing or over-writing a logical record are the same as for sequential files. Index integrity can be destroyed by rewriting records of different lengths. The user must position the file properly before each record is rewritten. Otherwise, writing takes place at the current position. Subsequent rewriting operations will rewrite the next record in the file, which is not necessarily the next index entry for the file.

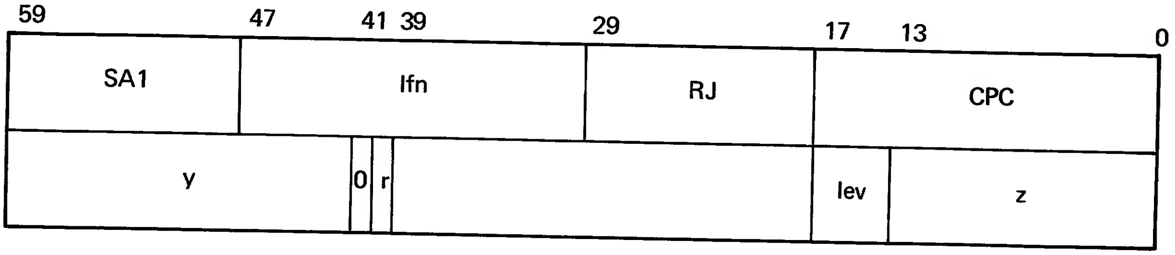
To position a random file for rewriting, the user may use one of two methods:

Set up the FET the same as for a random read and insert the record address found by searching the file index into the record request/return field in the seventh word of the FET.

For a SCOPE indexed file with records identified by name or number. use the WRITIN function. which causes the system to search the user's index and set the necessary FET fields.

Once the file is positioned to the beginning of a record, a REWRITE and REWRITER sequence or a WRITIN and REWRITER sequence can be executed without further repositioning. The record request/return field in the FET will be cleared by the first REWRITE or REWRITER that is issued by the calling program or WRITIN, and remain cleared until repositioning for another record is required.

The rewrite functions generate the following code:



The y and z fields depend on the specific function. The value of y is:

002 for REWRITE
003 for REWRITER or REWRITEF

The value of z is:

214 for REWRITE
224 for REWRITER
234 for REWRITEF

WRITIN FUNCTION

The WRITIN function applicable to mass storage files is a rewrite-in-place function similar to the rewrites. It assumes the user has full knowledge of file structure and knows the results of his actions, as explained with the rewrite functions.

WRITIN is similar to WRITOUT in that it relieves the user of the responsibility of manipulating buffer pointers when a working storage area is provided. When the circular buffer has been filled from the working storage area, WRITIN issues a REWRITE. Handling of binary and coded data is the same as for a WRITOUT. Parameters for WRITIN, and results of its use, are the same as for WRITOUT.

```
WRITIN  lfn

WRITIN  lfn,/name/

WRITIN  lfn,n
```

REWRITER is required to terminate a record, except when WRITIN or WRITOUT names another SCOPE indexed record. In this case a REWRITER of level 0 is forced before the new record is begun.

If a working storage area does not exist when WRITIN is issued to a random or sequential file, the function is ignored with no error indication. For a SCOPE indexed file without a working storage area, however, a WRITIN specifying a record name or number causes file repositioning to the beginning of that record. Therefore, the WRITIN function is useful before REWRITE or REWRITER.

The code generated by the WRITIN macro depends on the second parameter. For no parameter, a name parameter, and a number parameter, respectively, the code is:

59	29	17 0
	RJ	IOREWRT
		lfn

59	29	17 0
	RJ	IORRW
		lfn
name		

59	29	17 00
	RJ	IORRW
		lfn
		n

POSITIONING REQUESTS

Files can be repositioned forward with the SKIPF function, or repositioned in a reverse direction with BKSP, BKSPRU, REWIND, SKIPB, and UNLOAD. Any of these commands can be issued at any point in a logical record. If parity errors occur during repositioning, they will be ignored.

SKIPF	Skips records forward
SKIPB	Skips records backward
BKSP	Skips back single record
BKSPRU	Skips back single physical record unit
REWIND	Skips back to beginning-of-information
UNLOAD	Skips back to beginning-of-information and unloads

Reverse functions other than REWIND will stop at the beginning of the current reel of magnetic tape. No status returned to the FET indicates that beginning-of-reel has been detected before the requested number of backspaces was completed. However, if the XP bit (bit 40 of word 2) is set, the number of skips to be made will be stored in the RSC field (bits 24-42) of the FET extension.

If a magnetic file is positioned immediately after a newly written record when a reverse motion functions is issued, trailer label procedures will be executed before the function is performed. Four tape marks will be written if a trailer label format is not defined.

SKIPF FUNCTION

SKIPF causes one or more SCOPE logical records to be bypassed in a forward direction.

```
   SKIPF  lfn,n,lev,recall
```

The number of SCOPE logical records or record groups to be skipped is specified by the n parameter; the value 1 is assumed if n is absent. The maximum octal value of n is 777776. If n is 777777 and the file is on magnetic tape, it is not repositioned. If n is 777777 and the file is on mass storage, it is positioned at end-of-information.

The skip count is incremented as each level defined by the lev parameter is passed. Thus, a SKIPF with a count of 1 and lev of 0 issued in the middle of a record positions the file to the beginning of the following record.

The lev parameter specifies the level defining the record end: logical records are skipped until an end-of-record with a level number greater than or equal to the requested level is reached. The file is positioned immediately following this end-of-record mark.

If lev is absent, this field is set to zero, and the file is positioned forward n logical records or parts of records. If end-of-information is encountered before an end-of-record with the specified level is found, the end-of-information status bit will be set in the FET.

Although level numbers do not exist on S and L data format tapes, an lev parameter may be specified for SKIPF requests. If level number 17 is specified, a skip to end-of-file is performed. Any other level number is assumed to be zero, and one record is skipped.

On magnetic tape files, a SKIPF is continued across reels when the user processing (UP) bit is 0. If UP is set, the forward skip stops when end-of-reel is detected. If both UP and XP are set when end-of-reel appears before the skip count is fulfilled, the difference between the count requested and count made to that point will be returned to the RSC field in the FET extension.

The SKIPF macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th></th><th>39</th><th>29</th><th>17</th><th>13 00</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td colspan="2">CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2">n</td><td>lev</td><td>000240</td></tr>
</table>

SKIPB FUNCTION

SKIPB causes one or more SCOPE logical records to be bypassed in a reverse direction.

```
SKIPB  lfn,n,lev,recall
```

The number of SCOPE logical records or logical record groups to be skipped is specified by the n parameter; the value 1 is assumed if n is absent. When n is the maximum value of 777777 (octal), the file is rewound.

If the level parameter is used, logical records are read backwards until a short PRU containing the specified level has been read. A forward read is issued, leaving the file positioned after this short PRU. If the file is positioned initially between logical records, the level number immediately preceding the current position is ignored in searching for a SCOPE logical record of the specified level. This positioning process is performed n times.

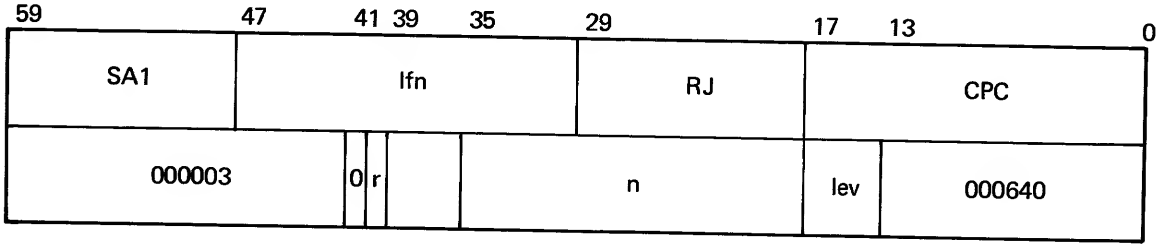
Consecutive SCOPE logical records within a file may be organized into a group by using level number. The file will be composed of one or more groups of logical records. This may be done by choosing a minimum level number other than 0, assigning a larger or equal level number to the last logical record of each group, and assigning a smaller level number to all other logical records. Then SKIPB lfn,,lev will skip the file backward to the beginning of the logical record group which immediately follows a logical record of level lev.

If the level parameter is absent, this field is set to zero, and the file is positioned backward n logical records (or partial logical records if the SKIPB is issued in the middle of a logical record).

If the load point is encountered before the requested level number is found, the request terminates with no indication. However, if XP is set, field RSC in the FET extension will contain the count n still required to complete the operation. Parity errors encountered during a SKIPB operation are ignored.

For S and L tapes, only levels 0 and 17 are recognized; any other level specified will be assumed 0.

The SKIPB macro generates the following code:

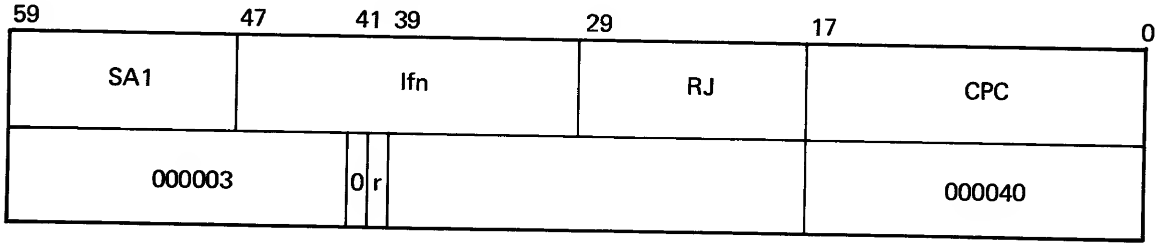


BKSP FUNCTION

BKSP causes one SCOPE logical record to be bypassed in a reverse direction. This function is a subset of SKIPB; it is included for compatibility with previous systems.

```
BKSP  lfn,recall
```

The BKSP macro generates the following code:



BKSPRU FUNCTION

BKSPRU causes one or more physical record units to be bypassed in a reverse direction.

```
BKSPRU  lfn,n,recall
```

The number of PRU's to be bypassed is indicated by n. If n does not appear, one PRU is skipped.

The BKSPRU macro generates the following code:

<table>
<tr><td>59</td><td>47</td><td>41</td><td></td><td>39</td><td>35</td><td>29</td><td>17 0</td></tr>
<tr><td>SA1</td><td colspan="5">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td></td><td colspan="2">n</td><td>000044</td></tr>
</table>

REWIND FUNCTION

REWIND positions a file to beginning-of-information. A REWIND issued for a file already rewound has no effect. A REWIND request for a file on a device that cannot be rewound causes a 22 status indicating an illegal function to be returned to the FET.

```
REWIND  lfn,recall
```

Labeled tapes will be positioned to beginning-of-information ahead of the label group. Subsequent forward motion requests will result in the label being skipped before the tape is read or written.

For unlabeled multi-reel tapes, a REWIND causes the current reel to be rewound. For labeled multi-reel, single-file tapes, a REWIND causes the current reel to be rewound and the volume number in the system tables to be set to 1. A subsequent forward motion will cause the label to be read and compared with the system tables, and the operator will be notified if the current reel is not number 1.

For multi-file labeled tapes, a REWIND issued for a file will cause positioning to the beginning of that file. If necessary, the operator will be instructed to mount the previous reel. A REWIND that references a multi-file name is illegal; the job will terminate.

For sequential pack files, a REWIND results in positioning to beginning-of-information also. with the operator instructed to mount a previous pack if necessary.

The REWIND macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th></th><th>39</th><th>29</th><th>17 0</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>000050</td></tr>
</table>

UNLOAD FUNCTION

UNLOAD operates in a manner similar to REWIND. except that it only affects the current reel of tape. UNLOAD cannot override an IU inhibit unload parameter on a REQUEST card. Otherwise. a tape file is rewound and unloaded.

```
UNLOAD  lfn,recall
```

The UNLOAD macro generates the following code:

<table>
<tr><th>59</th><th>47</th><th>41</th><th></th><th>39</th><th>29</th><th>17 0</th></tr>
<tr><td>SA1</td><td colspan="4">lfn</td><td>RJ</td><td>CPC</td></tr>
<tr><td colspan="2">000003</td><td>0</td><td>r</td><td colspan="2"></td><td>000060</td></tr>
</table>

FILE DISPOSITION

Files can be disposed of in two ways before job termination.

The file can be destroyed by the EVICT function.

The file can be routed to an output device at the central site or a remote terminal station with the DISPOSE function.

Files on public mass storage devices that have not been named in a DISPOSE control card or macro, or have not been equated to standard output file names such as OUTPUT or PUNCH, will disappear upon job termination. Permanent files, of course, will be retained under permanent file manager disposition.

It is not possible to dispose of a file by setting a disposition code directly in the FET.

EVICT FUNCTION

The EVICT function declares that contents of file lfn are to be discarded.

```
EVICT  lfn,recall
```

When a file on a public device is evicted, all space occupied by that file is released to the system. The space immediately becomes available for any system purpose or reassignment. An EVICT function directed to a permanent file is ignored; a dayfile message is issued and the job continues normally.

When a file on a magnetic tape is evicted, the tape is rewound and set to new status, thus declaring that the data and label are no longer valid and cannot be read by the job. If the file was declared to be labeled, a new header label will be written on any subsequent file reference. However, the evicted file will not be overwritten without operator authorization if the file expiration date has not passed.

If an EVICT function is directed to a member of a multi-file set, the set already must have been positioned at that file. Eviction of a member file also implies eviction of all files occupying higher numbered positions.

The logical file name used in the EVICT function is retained and cannot be used for a file on another device.

EVICT is undefined and, therefore, illegal on unit record equipment. A fatal error results if it is tried.

The EVICT macro generates the following code:

59	47	41		39	29	17 0
SA1	lfn				RJ	CPC
000003		0	r			000114

DISPOSE FUNCTION

With the DISPOSE function, a central processor program may declare a disposition code and initiate termination processing for a file. Files either can be released or sent to the output queue of completed files, as explained with the DISPOSE control card.

```
DISPOSE  lfn,*x=ky,recall
```

lfn   Logical file name

*   Optional end-of-job disposition indicator

x   Two-character disposition mnemonic. Only the first six codes have standard SCOPE 3.4 drivers.

PR	Print on 501 or 512 printer
P1	Print on 501 printer
P2	Print on 512 printer
PB	Punch binary
PU	Punch coded
P8	Punch 80 columns
FR	Print on microfilm recorder
FL	Plot on microfilm recorder
PT	Plot
HR	Print on hardcopy device
HL	Plot on hardcopy device

k   Optional site indicator; y must follow

C	Central site
I	INTERCOM terminal

y   Qualifier to k; y cannot be used without k

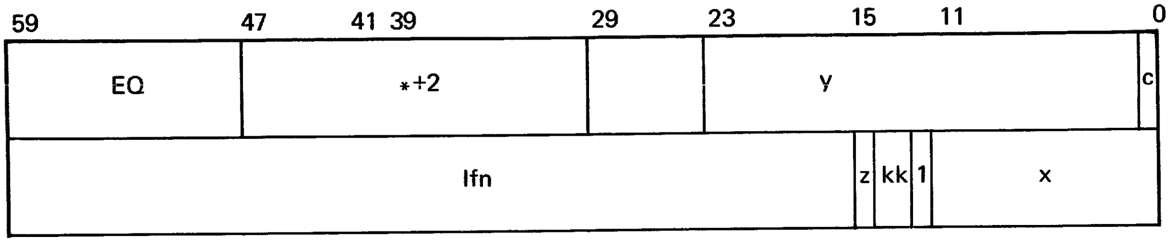
If k is C, two-character alphanumeric installation defined identifier of special forms or paper

If k is I, two-character user identification

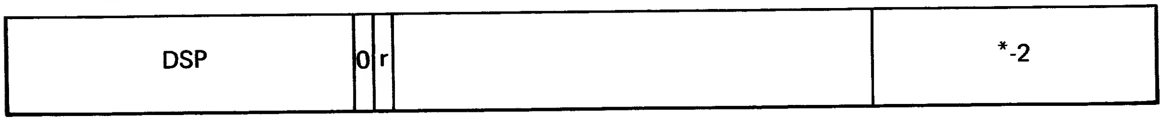
recall   Optional non-blank character indicating recall

If only lfn is given, the file is released, with mass storage and table references being removed.

The code generated by the DISPOSE macro is:



Form in X6:



RJ SYS=

z set to 1 when * is used

kk site indicator

00	none
01	central site
10	INTERCOM terminal

The completion bit is set to 1 by DSP when the requested function is complete.

SYSTEM TEXTS

System texts provide commonly used macro, micro, and symbol definitions for use in COMPASS source programs. SCOPE provides several system text overlays, which are loaded by COMPASS from the system libraries when specified by S parameters on the COMPASS control statement. S parameters can also be used on FTN control statements when FORTRAN source programs contain intermixed COMPASS subprograms. Up to seven system texts can be specified, each by a different S parameter, for a given assembler run. The system texts are made up of UPDATE common decks described below.

COMMON DECKS

System Action Request Macros: ACTCOM

IXi Xj*Xk
IXi Xj/Xk
IXi Xj/Xk,Bn
ABORT
CHECKPT
CLOCK
CONTRLC
DATE
DISPOSE
ENDRUN
FILESTAT
JDATE
LOADREQ
MEMORY
MESSAGE
RECALL
RECOVR
REQUEST
RTIME
SYSCOM
SYSTEM
TIME
TRANSR

Input/Output Macros using CPC: CPSYS

BKSP
BKSPRU
CLOSE
CLOSER
EVICT
FILEB
FILEC
LABEL
OPEN
POSMF
READ
READC
READIN
READN
READNS
READSKP
REWIND
REWRITE
REWRITEF
REWRITER
RFILEB
RFILEC
RPHR
SKIPB
SKIPF
UNLOAD
WPHR
WRITE
WRITEC
WRITEN
WRITEF
WRITER
WRITIN
WRITOUT

Record Manager Internal Text: CGRMITXT

Contains macro, micro, and symbol definitions used within Record Manager modules.

Installation Parameters: IPARAMS

Contains installation parameters as symbol and micro definitions.

Loader Request Macros: LMACOM

Contains two macros: LOADER and LDREQ.

Permanent File Macros: PFCOM

ALTER
ATTACH
CATALOG
EXTEND
FDB
PERM
PURGE
RENAME
SETP

Peripheral Processor System Definitions: PPSYS

Contains many system symbols and micros, and the following macros:

ADK
BIT
CEQU
CMICRO
CRI
ENM
JOBCARD
LDCA
LDK
PPENTRY
SBK
UJK

Integrated Scheduler Macros: SCHCOM

CISO
ENTRY34
LDW
SCHLOK
SCHSAVE
SCHSTOR
STREQ

Indexed Sequential Interface Macros: SISICOM

ACCESSK
ACCESSN
DELETE
FORCEW
INSERT
OPENNEW
OPENOLD
REPLACE
REPOS
SEEKL
SEEKS
SETBLKD
SETBLKI
SETCOLL
SETERR
SETFET
SETKEY
TERMNAT

Record Manager Definitions: 6RMCOM

Contains macro, micro, and symbol definitions for user programs that use the Record Manager.

TEXT OVERLAYS

The SCOPE system text overlays contain various combinations of the common decks, as shown below.

CPCTEXT

System text for central processor programs using CPC.

Common decks ACTCOM, CPSYS, and SISICOM.

IOTEXT

System text for central processor programs using Record Manager.

Common decks ACTCOM and 6RMCOM.

IPTEXT

Installation parameter system text.

Contains a single macro, IPARAMS, whose body is the IPARAMS common deck.

LDRTEXT

System text for central processor programs using loader.

Common deck LMACOM.

PFMTEXT

System text for central processor programs using permanent files.

Common deck PFCOM.

PPTEXT

System text for peripheral processor programs.

Common deck PPSYS.

SCHTEXT

System text for central and peripheral processor programs interfacing with the integerated scheduler.

Common deck SCHCOM.

SCPTEXT

System text for central and peripheral processor programs in SCOPE.

Common decks ACTCOM, CPSYS, and PPSYS.

SYSTEXT

System text for central processor programs.

This is the default system text used by COMPASS when no S or G parameters are specified. It can be identical to either CPCTEXT or IOTEXT, at installation option. In the released system, SYSTEXT is equal to IOTEXT.

TXT6RM

System text for Record Manager modules.

Common decks ACTCOM and CGRMITXT.

In addition to the above system texts provided by SCOPE, the following system texts are provided by product set members.

ALGTEXT

Contains COMPASS coded macros used to expand application areas of ALGOL-60.

FTNMAC

Contains macros used by COMPASS object programs produced by the FORTRAN Extended compiler (FTN).

SMTEXT

Contains macros for central processor programs that call the SORT/MERGE system.

STANDARD SCOPE CHARACTER SETS A

The character set selected when the system is installed should be compatible with the printers.

With an installation parameter, the installation keypunch format standard can be selected as 026 or 029; the installation parameter can also allow a user to override the standard; a user may select a keypunch mode for his input deck by punching 26 or 29 in columns 79 and 80 of his JOB card or any 7/8/9 end-of-record card. The mode remains set for the remainder of the job or until it is reset by a different mode selection on another 7/8/9 card.

The physical end of a unit record (card, print line or any other significant character string) is delimited by 12 zero bits. The use of two colons in succession will produce 12 zero bits interpreted as a unit record terminator. This general restriction on the use of the colon character applies throughout SCOPE 3 and all of its product sets.

SCOPE 3.4

STANDARD CHARACTER SETS

CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code
:†	:	00†	8-2	00	8-2	3A
A	A	01	12-1	61	12-1	41
B	B	02	12-2	62	12-2	42
C	C	03	12-3	63	12-3	43
D	D	04	12-4	64	12-4	44
E	E	05	12-5	65	12-5	45
F	F	06	12-6	66	12-6	46
G	G	07	12-7	67	12-7	47
H	H	10	12-8	70	12-8	48
I	I	11	12-9	71	12-9	49
J	J	12	11-1	41	11-1	4A
K	K	13	11-2	42	11-2	4B
L	L	14	11-3	43	11-3	4C
M	M	15	11-4	44	11-4	4D
N	N	16	11-5	45	11-5	4E
O	O	17	11-6	46	11-6	4F
P	P	20	11-7	47	11-7	50
Q	Q	21	11-8	50	11-8	51
R	R	22	11-9	51	11-9	52
S	S	23	0-2	22	0-2	53
T	T	24	0-3	23	0-3	54
U	U	25	0-4	24	0-4	55
V	V	26	0-5	25	0-5	56
W	W	27	0-6	26	0-6	57
X	X	30	0-7	27	0-7	58
Y	Y	31	0-8	30	0-8	59
Z	Z	32	0-9	31	0-9	5A
0	0	33	0	12	0	30
1	1	34	1	01	1	31
2	2	35	2	02	2	32
3	3	36	3	03	3	33
4	4	37	4	04	4	34
5	5	40	5	05	5	35
6	6	41	6	06	6	36
7	7	42	7	07	7	37
8	8	43	8	10	8	38
9	9	44	9	11	9	39
+	+	45	12	60	12-8-6	2B
–	–	46	11	40	11	2D
*	*	47	11-8-4	54	11-8-4	2A
/	/	50	0-1	21	0-1	2F
(	(	51	0-8-4	34	12-8-5	28
)	)	52	12-8-4	74	11-8-5	29
$	$	53	11-8-3	53	11-8-3	24
=	=	54	8-3	13	8-6	3D
blank	blank	55	no punch	20	no punch	20
, (comma)	, (comma)	56	0-8-3	33	0-8-3	2C
. (period)	. (period)	57	12-8-3	73	12-8-3	2E
≡	#	60	0-8-6	36	8-3	23
[	[	61	8-7	17	12-8-2	5B
]	]	62	0-8-2	32	11-8-2	5D
%††	%	63	8-6	16	0-8-4	25
≠	" (quote)	64	8-4	14	8-7	22
→	_ (underline)	65	0-8-5	35	0-8-5	5F
∨	!	66	11-0 or 11-8-2†††	52	12-8-7 or 11-0†††	21
∧	&	67	0-8-7	37	12	26
↑	' (apostrophe)	70	11-8-5	55	8-5	27
↓	?	71	11-8-6	56	0-8-7	3F
<	<	72	12-0 or 12-8-2†††	72	12-8-4 or 12-0†††	3C
>	>	73	11-8-7	57	0-8-6	3E
≤	@	74	8-5	15	8-4	40
≥	\	75	12-8-5	75	0-8-2	5C
¬	⌢(circumflex)	76	12-8-6	76	11-8-7	5E
; (semicolon)	; (semicolon)	77	12-8-7	77	11-8-6	3B

†Twelve or more zero bits at the end of a 60-bit word are interpreted as end-of-line mark rather than two colons. End-of-line mark is converted to external BCD 1632.

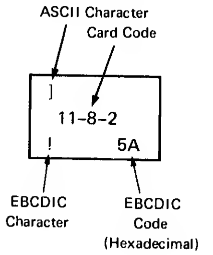
††In installations using the CDC 63-graphic set, display code 00 has no associated graphic or Hollerith code; display code 63 is the colon (8-2 punch).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) WITH PUNCHED CARD CODES AND EBCDIC TRANSLATION

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 0 F0	@ 8-4 @ 7C	P 11-7 P D7	8-1 ` 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 88	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SDH 01	DC1 11-9-1 DC1 11	! 12-8-7 | 4F	1 1 1 F1	A 12-1 A C1	O 11-8 O D8	a 12-0-1 a 81	q 12-11-8 q 98	0-9-1 SDS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 89	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	" 8-7 " 7F	2 2 2 F2	8 12-2 8 C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 CC 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 8A	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 TM 13	# 8-3 # 7B	3 3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 11-0-2 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 88	12-11-9-8-7 DF
0 1 0 0	4	EDT 9-7 EDT 37	DC4 9-8-4 DC4 3C	$ 11-8-3 $ 58	4 4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 11-0-3 t A3	0-9-4 8YP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 8C	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 11-0-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 88	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 E8
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 11-0-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 8E	11-0-9-8-4 ⑁ EC
0 1 1 1	7	8EL 0-9-8-7 8EL 2F	ETB 0-9-6 ET8 26	' 8-5 ' 7D	7 7 7 F7	G 12-7 G C7	W 0-6 . W E6	g 12-0-7 g 87	w 11-0-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 8F	11-0-9-8-5 ED
1 0 0 0	8	8S 11-9-6 8S 16	CAN 11-9-8 CAN 18	(12-8-5 (4D	8 8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 11-0-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 80	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19	) 11-8-5) 5D	9 9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 11-0-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 69	12-0-8-7 8F	12-11-0-1 81	12-0-9-8-3 C8	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SU8 9-8-7 SUB 3F	* 11-8-4 * 5C	: 8-2 : 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 11-0-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 B2	12-0-9-8-4 ⑀ CC	12-11-0-9-8-2 I(LVM) FA
1 0 1 1	11 (8)	VT 12-9-8-3 VT 08	ESC 0-9-7 ESC 27	+ 12-8-6 + 4E	; 11-8-6 ; 5E	K 11-2 K D2	[12-8-2 ¢ 4A	k 12-11-2 k 92	{ 12-0 { C0	0-9-8-3 CU2 2B	9-8-3 CU3 38	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 F8
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	, 0-8-3 , 68	< 12-8-4 < 4C	L 11-3 L D3	\ ' 0-8-2 \ E0	l 12-11-3 l 93	¦ 12-11 ¦ 6A	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 9B	12-11-0-4 B4	12-0-9-8-6 ⑂ CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	– 11 – 60	= 8-6 = 7E	M 11-4 M D4	] 11-8-2 ! 5A	m 12-11-4 m 94	} 11-0 } D0	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 B5	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SD 12-9-8-6 SO 0E	RS 11-9-8-6 IRS 1E	. 12-8-3 . 48	> 0-8-6 > 6E	N 11-5 N D5	^ 11-8-7 ¬ 5F	n 12-11-5 n 95	~ 11-0-1 ~ A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 86	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	/ 0-1 / 61	? 0-8-7 ? 6F	O 11-6 O D6	‾ 0-8-5 _ 6D	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 1B	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 87	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

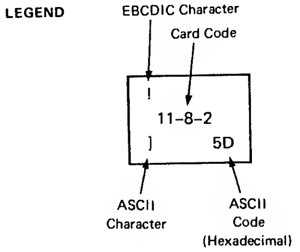
LEGEND



EXTENDED BINARY CODED DECIMAL INTERCHANGE CO⊃E (EBCDIC) WITH PUNCHED CARD CODES AND ASCII TRANSLATION

BITS 0 1 2 3		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
BITS 4 5 6 7	1ST HEX 2ND	0	1	2	3	4	5	6	7	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	DS 11-0-9-8-1 80	12-11-0-9-8-1 90	SP no punch SP 20	& 12 & 26	- 11 - 2D	12-11-0 8A	12-0 8-1 C3	12-11-8-1 CA	11-0 8-1 D1	12-11-0-8-1 D8	{ 12-0 { 7B	} 11-0 } 7D	\ 0-8-2 \ 5C	0 0 0 30
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	SOS 0 9 1 81	9-1 91	12-0-9-1 A0	12-11-9-1 A9	/ 0-1 / 2F	12-11-0-9-1 88	a 12-0-1 a 61	j 12-11-1 j 6A	~ 11-0-1 ~ 7E	12-11-0-1 D9	A 12-1 A 41	J 11-1 J 4A	11-0-9-1 9F	1 1 1 31
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	FS 0-9-2 82	SYN 9-2 SYN 16	12-0-9-2 A1	12-11-9-2 AA	11-0-9-2 82	12-11-0-9-2 8C	b 12-0 2 b 62	k 12-11-2 k 6B	s 11-0-2 s 73	12-11-0-2 DA	B 12-2 B 42	K 11-2 K 4B	S 0-2 S 53	2 2 2 32
0 0 1 1	3	ETX 12-9-3 ETX 03	TM 11-9-3 DC3 13	0-9-3 83	9-3 93	12-0-9-3 A2	12-11-9-3 A8	11-0-9-3 B3	12-11-0-9-3 8D	c 12-0-3 c 63	l 12-11-3 l 6C	t 11-0-3 t 74	12-11-0-3 DB	C 12-3 C 43	L 11-3 L 4C	T 0-3 T 54	3 3 3 33
0 1 0 0	4	PF 12-9-4 9C	RES 11-9-4 9D	BYP 0-9-4 84	PN 9-4 94	12-0-9-4 A3	12-11-9 4 AC	11-0-9-4 B4	12-11-0-9-4 BE	d 12-0 4 d 64	m 12-11-4 m 6D	u 11-0-4 u 75	12-11-0-4 DC	D 12-4 D 44	M 11-4 M 4D	U 0-4 U 55	4 4 4 34
0 1 0 1	5	HT 12-9-5 HT 09	NL 11 9-5 85	LF 0-9-5 LF 0A	RS 9-5 95	12-0-9-5 A4	12-11-9-5 AD	11-0-9-5 B5	12-11-0-9-5 BF	e 12-0-5 e 65	n 12-11-5 n 6E	v 11-0-5 v 76	12-11-0-5 DD	E 12-5 E 45	N 11-5 N 4E	V 0-5 V 56	5 5 5 35
0 1 1 0	6	LC 12-9-6 86	8S 11-9-6 BS 08	ETB 0-9-6 ETB 17	UC 9-6 96	12-0-9-6 A5	12-11-9-6 AE	11-0-9-6 86	12-11-0-9 6 C0	f 12-0 6 f 66	o 12-11-6 o 6F	w 11-0 6 w 77	12-11-0-6 DE	F 12-6 F 46	O 11-6 O 4F	W 0-6 W 57	6 6 6 36
0 1 1 1	7	DEL 12-9-7 DEL 7F	IL 11-9-7 87	ESC 0-9-7 ESC 1B	EOT 9-7 EDT 04	12-0-9-7 A6	12-11-9 7 AF	11-0-9-7 87	12-11-0-9-7 C1	g 12-0-7 g 67	p 12-11-7 p 70	x 11 0-7 x 78	12-11-0-7 DF	G 12-7 G 47	P 11-7 P 50	X 0-7 X 58	7 7 7 37
1 0 0 0	8	GE 12-9-8 97	CAN 11-9-8 CAN 18	0-9-8 88	9-8 98	12-0-9-8 A7	12-11 9 8 B0	11-0-9 8 88	12-11-0-9-8 C2	h 12-0-8 h 68	q 12-11-8 q 71	y 11-0 8 y 79	12-11-0-8 E0	H 12-8 H 48	O 11-8 Q 51	Y 0-8 Y 59	8 8 8 38
1 0 0 1	9	RLF 12-9-8-1 8D	EM 11-9-8-1 EM 19	0 9 8 1 89	9 8 1 99	12-8-1 A8	11-8-1 B1	0-8-1 B9	` 8-1 ` 60	i 12-0-9 i 69	r 12-11-9 r 72	z 11-0-9 z 7A	12-11-0-9 E1	I 12-9 I 49	R 11-9 R 52	Z 0-9 Z 5A	9 9 9 39
1 0 1 0	A (10)	SMM 12-9-8-2 8E	CC 11-9-8-2 92	SM 0-9-8-2 8A	9-8-2 9A	¢ 12-8-2 [58	! 11-8-2] 5D	¦ 12-1' ¦ 7C	: 8-2 : 3A	12-0-8-2 C4	12-11-8-2 CB	11-0-8-2 D2	12-11-0-8-2 E2	12-0-9-8-2 E8	12-11-9-8-2 EE	11-0-9-8-2 F4	|(LVM) 12-11-0-9-8-2 FA
1 0 1 1	B (11)	VT 12-9-8-3 VT 0B	CU1 11-9-8-3 8F	CU2 0-9-8-3 8B	CU3 9-8-3 98	. 12-8-3 2E	$ 11-8-3 $ 24	, 0-8-3 , 2C	# 8-3 # 23	12-0 8-3 C5	12-11-8-3 CC	11-0 8-3 D3	12-11-0-8-3 E3	12-0-9-8-3 E9	12-11-9-8-3 EF	11-0-9-8-3 F5	12-11-0-9-8-3 FB
1 1 0 0	C (12)	FF 12-9-8-4 FF 0C	IFS 11-9-8-4 FS 1C	0-9-8-4 8C	DC4 9-8-4 DC4 14	< 12-8-4 < 3C	* 11-8-4 * 2A	% 0-8-4 % 25	@ 8-4 @ 40	12-0-8-4 C6	12-11-8-4 CD	11-0-8-4 D4	12-11-0-8-4 E4	⑁ 12-0-9-8-4 EA	12-11-9-8-4 F0	⑀ 11-0-9-8-4 F6	12-11-0-9-8-4 FC
1 1 0 1	D (13)	CR 12-9-8-5 CR 0D	IGS 11-9-8-5 GS 1D	ENO 0-9-8-5 ENO 05	NAK 9-8-5 NAK 15	(12-8-5 (28	) 11-8-5) 29	_ 0-8-5 5F	' 8-5 ' 27	12 0-8-5 C7	12-11-8-5 CE	11-0-8-5 D5	12-11-0-8-5 E5	12-0-9-8-5 EB	12-11-9-8-5 F1	11-0-9-8-5 F7	12-11-0-9-8-5 FD
1 1 1 0	E (14)	SO 12-9-8-6 SO 0E	IRS 11-9-8-6 RS 1E	ACK 0-9-8-6 ACK 06	9-8-6 9E	+ 12-8-6 + 28	; 11-8-6 ; 3B	> 0-8-6 > 3E	= 8-6 3D	12-0-8-6 C8	12-11-8-6 CF	11-0-8-6 D6	12-11-0-8-6 E6	⑂ 12-0-9-8-6 EC	12-11-9-8-6 F2	11-0-9-8-6 F8	12-11-0-9-8-6 FE
1 1 1 1	F (15)	SI 12-9-8-7 SI 0F	IUS 11-9-8-7 US 1F	8EL 0-9-8-7 8EL 07	SUB 9-8-7 SUB 1A	| 12-8-7 ! 21	¬ 11-8-7 ^ 5E	? 0-8-7 ? 3F	" 8-7 " 22	12-0-8-7 C9	12-11-8-7 D0	11-0-8-7 D7	12-11-0-8-7 E7	12-0-9-8-7 ED	12-11-9-8-7 F3	11-0-9-8-7 F9	12-11-0-9-8-7 EO FF

LEGEND



CONTROL DATA CHARACTER SETS
SHOWING TRANSLATIONS BETWEEN DISPLAY CODE AND ASCII/EBCDIC

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
00	:	:	3A	SUB	1A	:	7A	SUB	3F
01	A	A	41	a	61	A	C1	a	81
02	B	B	42	b	62	B	C2	b	82
03	C	C	43	c	63	C	C3	c	83
04	D	D	44	d	64	D	C4	d	84
05	E	E	45	e	65	E	C5	e	85
06	F	F	46	f	66	F	C6	f	86
07	G	G	47	g	67	G	C7	g	87
10	H	H	48	h	68	H	C8	h	88
11	I	I	49	i	69	I	C9	i	89
12	J	J	4A	j	6A	J	D1	j	91
13	K	K	4B	k	6B	K	D2	k	92
14	L	L	4C	l	6C	L	D3	l	93
15	M	M	4D	m	6D	M	D4	m	94
16	N	N	4E	n	6E	N	D5	n	95
17	O	O	4F	o	6F	O	D6	o	96
20	P	P	50	p	70	P	D7	p	97
21	O	Q	51	q	71	O	D8	q	98
22	R	R	52	r	72	R	D9	r	99
23	S	S	53	s	73	S	E2	s	A2
24	T	T	54	t	74	T	E3	t	A3
25	U	U	55	u	75	U	E4	u	A4
26	V	V	56	v	76	V	E5	v	A5
27	W	W	57	w	77	W	E6	w	A6
30	X	X	58	x	78	X	E7	x	A7
31	Y	Y	59	y	79	Y	E8	y	A8
32	Z	Z	5A	z	7A	Z	E9	z	A9
33	0	0	30	DLE	10	0	F0	DLE	10
34	1	1	31	DC1	11	1	F1	DC1	11
35	2	2	32	DC2	12	2	F2	DC2	12
36	3	3	33	DC3	13	3	F3	TM	13
37	4	4	34	DC4	14	4	F4	DC4	3C
40	5	5	35	NAK	15	5	F5	NAK	3D
41	6	6	36	SYN	16	6	F6	SYN	32
42	7	7	37	ETB	17	7	F7	ETB	26
43	8	8	38	CAN	18	8	F8	CAN	18
44	9	9	39	EM	19	9	F9	EM	19
45	+	+	2B	VT	0B	+	4E	VT	0B
46	–	–	2D	CR	0D	–	60	CR	0D
47	*	*	2A	LF	0A	*	5C	LF	25
50	/	/	2F	SI	0F	/	61	SI	0F
51	(	(	28	BS	08	(	4D	BS	16
52	)	)	29	HT	09	)	5D	HT	05
53	S	S	24	EOT	04	S	5B	EOT	37
54	=	=	3D	GS	1D	=	7E	IGS	1D
55	SP	SP	20	NUL	00	SP	40	NUL	00
56	,	,	2C	FF	0C	,	6B	FF	0C
57	.	.	2E	SO	0E	.	4B	SO	0E
60	≡ =	=	23	ETX	03	=	7B	ETX	03
61	[	[	5B	FS	1C	c	4A	IFS	1C
62	]	]	5D	SOH	01	!	5A	SOH	01
63	%	%	25	ENO	05	%	6C	ENO	2D
64	≠ "	"	22	STX	02	"	7F	STX	02
65	→ _	_	5F	DEL	7F	_	6D	DEL	07
66	∨ !	!	21	}	7D	|	4F	} ,	D0
67	∧ &	&	26	ACK	06	&	50	ACK	2E
70	↑ '	'	27	BEL	07	'	7D	BEL	2F
71	↓ ?	?	3F	US	1F	?	6F	IUS	1F
72	<	<	3C	{	7B	<	4C	{	C0
73	>	>	3E	RS	1E	>	6E	IRS	1E
74	≤ @	@	40	`	60	@	7C	`	79
75	≥ \	\	5C	¦	7C	\	E0	¦	6A
76	¬ ^	^	5E	~	7E	¬	5F	~	A1
77	;	;	3B	ESC	1B	;	5E	ESC	27

NOTES:

1. The terms "upper case" and "lower case" apply only to the case conversions, and do not necessarily reflect any true "case".
2. When translating from Display Code to ASCII/EBCDIC, the "upper case" equivalent character is taken.
3. When translating from ASCII/EBCDIC to Display Code, the "upper case" and "lower case" characters fold together to a single Display Code equivalent character.
4. All ASCII and EBCDIC codes not listed are translated to Display Code 55 (SP).
5. Where two Display Code graphics are shown for a single octal code, the leftmost graphic corresponds to the CDC 64-character set (system assembled with IP.CSET set to C64.1), and the rightmost graphic corresponds to the CDC 64-character ASCII subset (system assembled with IP.CSET set to C64.2).
6. In a 63-character set system, the display code for the : graphic is 63. The % character does not exist, and translations from ASCII/EBCDIC % or ENO yield blank (55_8).

HEXADECIMAL–OCTAL CONVERSION TABLE

		First Hexadecimal Digit															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Second Hexadecimal Digit	0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
	1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
	2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
	3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
	4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
	5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
	6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
	7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
	8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
	9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
	A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
	B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
	C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
	D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
	E	016	036	056	076	116	136	156	176	216	236	256	276	316	336	356	376
	F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377
Octal		000 – 037		040 – 077		100 – 137		140 – 177		200 – 237		240 – 277		300 – 337		340 – 377	

CARD FORMATS B

SCOPE components can process three types of cards: Hollerith, standard binary and free-form binary. Hollerith cards are often called coded cards. Each column can be punched to represent codes of any given character set (Appendix A). The hole code is translated by card reading devices into the binary code for the character. Blank columns are translated into a binary code representing a blank or space.

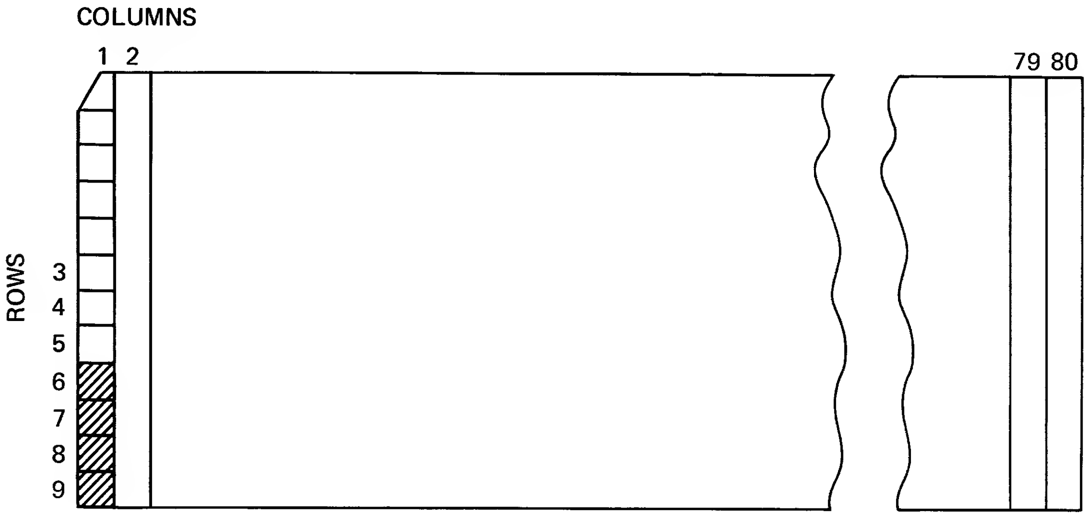
Binary cards contain 12 binary digits in each column; a punch represents a 1, the absence of a punch represents 0. Binary cards are not translated by card reading devices. The several types of standard binary cards are differentiated from Hollerith cards by a special code in column 1 (the leftmost column). The binary digits are read in sequence from top to bottom of the column; the columns are read from left to right.

All standard binary cards must have punches in rows 7 and 9 of column 1; thus, any four octal digits ending with 5 or 7 would act as a binary card marker. Any card without a 7/9 punch in column 1 is considered to be a Hollerith card; no legal Hollerith code contains a 7/9 punch combination. Any Hollerith card column containing an illegal Hollerith punch combination is read as a blank, and a message is produced for output giving the card number and the number of the record containing the card.

Several types of binary cards are considered standard by SCOPE: binary subprogram or data cards, end-of-file cards and end-of-record cards.

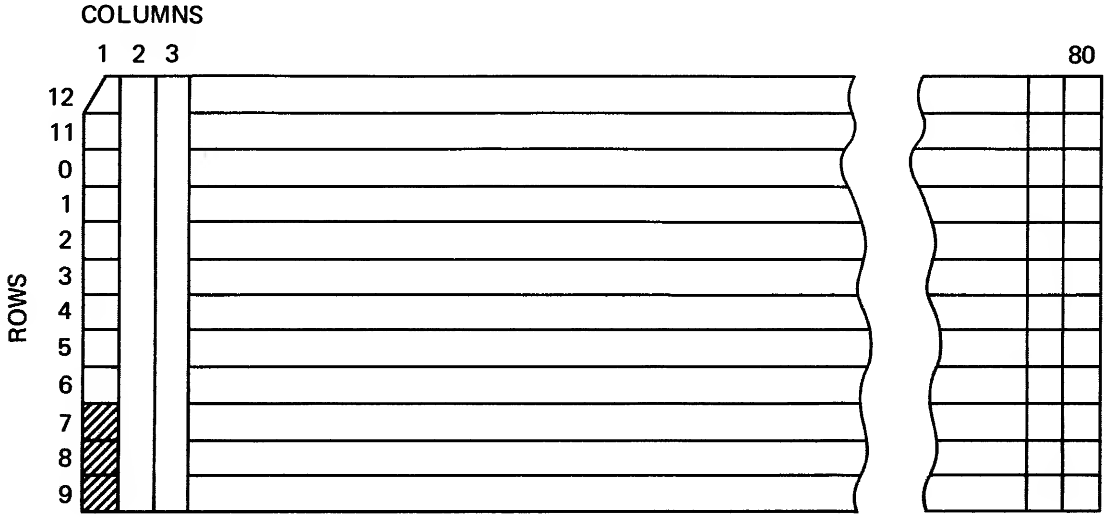
END-OF-FILE CARD

An end-of-file card contains octal 0017 in column 1. This card, commonly referred to as a 6/7/8/9 card, is shown below:



END-OF-RECORD CARD

A card containing octal 0007 (7/8/9) in column 1 is recognized as an end-of-record marker. Level numbers associated with the record are punched in Hollerith code in columns 2 and 3. The level number may be 00,01,02,03,04,05,06,07,10,11,12,13,14,15,16, or 17. If columns 2 and 3 are blank, the level number is assumed to be 00. Level numbers 1-7 may be punched with a trailing blank in the form nb where n is the level number and b is a blank. The format of an EOR card follows:

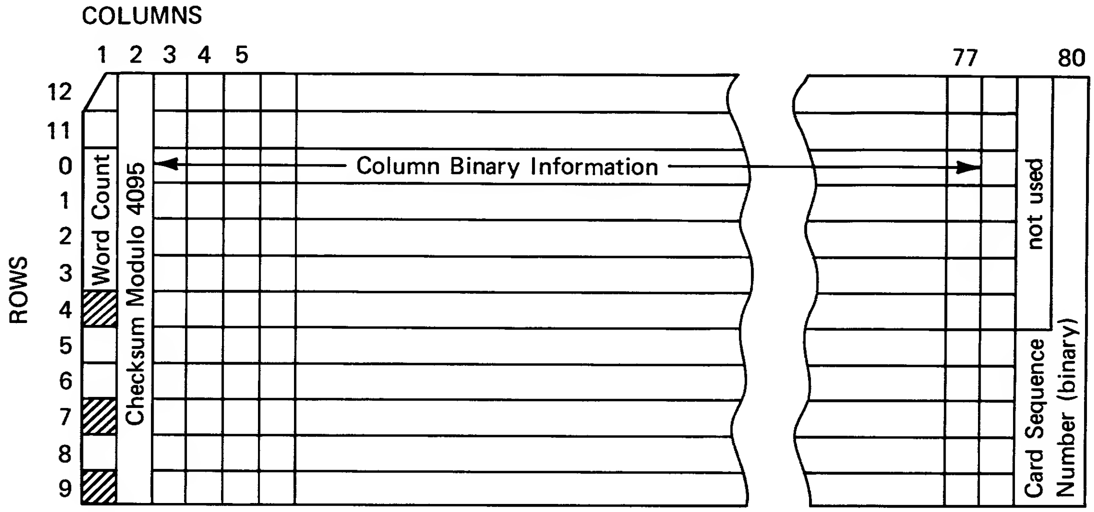


BINARY SUBPROGRAM CARDS

Binary subprogram or data cards can contain the binary representation of up to 15 central memory words. This card type contains a 7/9 punch along with a word count in rows 0/1/2/3, and a checksum flag in row 4, all in column 1. The word count indicates the number of binary words in the card, starting in column 3 and not extending beyond column 77. Column 2 contains a checksum of the binary words in columns 3 thru 77; the value of the checksum is a ones-complement sum, modulo 4095 (2**12-1=4095). If the checksum flag in row 4 of column 1 is punched, the checksum is ignored by the system. Columns 79 and 80 contain a card sequence number in binary. The lower 5 bits in column 79 and all 12 bits in column 80 are used to make up the 17-bit serial number of the card record within the logical record that contains it. If cards are not read in sequential order, a warning message is produced for output; however, the cards are read and accepted.

Columns 1, 2, 78, 79, and 80 are produced when a binary punch file is punched through a remote terminal or JANUS controlled device. These columns are removed when the deck is read into the system, such that a card has only 15 central memory words of information internally.

The format of a binary subprogram card is shown below:



FREE-FORM BINARY CARDS

Free-form binary cards are unique in that they can be read as sixteen 60-bit words per card (eighty 12 bit columns). They contain no checksum or sequence number. As an example, a card having 6/7/8/9 punched in column 1, and having at least one punch in one other column, can be read as a free-form binary card. Normally, it would be treated as an end-of-file card.

Free-form binary cards must be set up in a special logical record having the following format: The first card must have all 12 rows punched in both columns 1 and 2; the card must not contain any other punches. This flag card is not read as containing information; it signals that free-form binary cards follow.

Any number of cards may follow; none may have the same form as the free-form flag card described above, and none may have the same form as a 6/7/8/9 end-of-file card. The free-form binary cards are read into memory in 16-word increments. After the free-form binary cards, another flag card with all 12 rows punched in columns 1 and 2 must appear. This card signals the end of the free-form binary deck, and that standard binary or Hollerith cards will follow.

If it is necessary for a free-form binary card with the same appearance of the flag card to appear in the deck, it is possible to create a flag card of a slightly different form. Any card having all 12 rows in column 1 punched and all 12 rows in any other column punched with no other punches on the card is recognized as a free-form flag; therefore, 79 variations are possible for the flag card.

Normally, a series of free-form binary cards and their flag cards are organized into one record in an input file. However, they may be preceded and/or followed by standard binary and/or Hollerith cards within the same record. The different cards in the record are accepted; however, a message indicating a change in mode is produced for the record. A valid record might consist of the following:

1. A series of Hollerith cards.

2. A start free-form flag card (7777 in both column 1 and 80) with no other punches.

3. A series of free-form binary cards not including a standard end-of-file card, nor any identical with 2.

4. An end-free-form flag card, identical with 2.

5. A start-free-form flag card, which might be the same as or different from 2 and 4.

6. A series of free-form binary cards, as in 3.

7. An end-free-form flag card, identical with 5.

8. A series of standard binary cards which should be in order according to sequence numbers. If not, a sequence number check message will be issued for the record. A mode change message will be issued for the record.

9. An end-of-record (7/8/9) card.

Free-form binary cards are produced when the disposition code for the output punch file is P8 (octal 0014). If the number of words to be punched in free-form is not an even multiple of 16, the unused columns at the right of the last punched card will be blank. A standard end-of-record card will be produced following the last free-form binary card. The flag cards are not punched as part of the output.

Standard binary cards are produced when the file name is PUNCHB or the file has a disposition code of PB (octal 0012); Hollerith cards are produced when the file name is PUNCH or the disposition code of the output file is PU (octal 0010). Unused columns at the end of the last Hollerith card will be blank; an end-of-record card will follow the last card produced.

CONTROL CARD SUMMARY C

All requests that can be issued to SCOPE on control cards are noted in the following summary, together with parameters issued as part of each request. In this summary, constants are capitalized and variables are in lower case; the variables are defined below the illustrated format. In the formats shown, commas are used as separators and periods are used as terminators in all cases.

Following the description of each request, the function executed by that request and the document or reference manual (RM) that contains a full discussion of that function are noted.

Required characteristics of parameters appearing frequently are:

lfn	Logical file names of 1-7 characters beginning with a letter
pname	Pack family name of 1-7 characters beginning with a letter
vsn	Volume serial number of 1-6 characters
lev	Octal level number 1-17 for SCOPE logical records

Card Format	Function	Reference Section
JOB IDENTIFICATION AND CONTROL CARDS		
job,Tt,CMa,ECb,Pp,Dym,MTn,TPn,NTn,CPcp,RT. job Job name t CP time limit a Central memory storage b ECS field length p Priority level y Letters identifying dependent job string m Number of jobs on which this depends n Number of tape units required; MT or TP (7 track), NT (9 track) cp Central processor type A or 66, B or 64	Identifies job and specifies system requirements	SCOPE 3.4 RM, 4
RFL,nfl. nfl New field length	Redefines job field length	SCOPE 3.4 RM, 4
TRANSF,lfn1,lfn2,...lfnn. lfn Name of next job for execution	Decreases dependency count of named jobs	SCOPE 3.4 RM, 4
LIMIT,n. n Octal number representing the multiple of 10000 60-bit words	Limits amount of mass storage assigned to job	SCOPE 3.4 RM, 4

Card Format	Function	Reference Section
EQUIPMENT AND FILE ASSIGNMENT CARDS		

DISPOSE(lfn,x)
DISPOSE(lfn,x=ky)
	Releases files to output device	SCOPE 3.4 RM, 4

DISPOSE(lfn,*x=ky)	Dispose file at EOJ; ignores ky	SCOPE 3.4 RM, 4
lfn File to be released x Disposition code ky k=C (central site) y must be 2-character device code defined bv installation; k=I (INTERCOM) y designates user ID		
REMOVE,lfn,pname. lfn Name of file pname Family pack name	Removes files from family pack	SCOPE 3.4 RM, 4
REQUEST,lfn,DP. REQUEST,lfn,2DP. lfn Logical file name	Assigns one (DP or two (2DP) disk packs on which new output file, lfn, is written	SCOPE 3.4 RM, 4
REQUEST,lfn,DP,E,VID=vid. vid Visual identifier	Assigns sequential disk pack which contains existing files	SCOPE 3.4 RM, 4
REQUEST,lfn,dt,eq. lfn Logical file referenced dt Device or device type; dt field may be expanded eq Specific device (EST ordinal)	Requests assigment of device on which file resides or will be written	SCOPE 3.4 RM, 4

Card Format	Function	Reference Section
REQUEST,lfn,PK,pname. pname Pack family name	Requests file be assigned to family pack	SCOPE 3.4 RM, 4
RETURN,lfn1,lfn2,...,lfnn. lfn Logical file names	Releases files and associated devices from job and decrements tape unit required count	SCOPE 3.4 RM, 4
RPACK,pname,E,vsn. pname Pack family name vsn Volume serial number	Assigns family disk pack which contains existing files	SCOPE 3.4 RM, 4
RPACK,pname,N. pname Pack family name	Assigns family disk pack on which new (N) output file is written	SCOPE 3.4 RM, 4
UNLOAD,lfn1,lfn2,...,lfnn. lfn Logical file names	Same as RETURN but does not decrement tape unit count	SCOPE 3.4 RM, 4
LOADER CONTROL CARDS		
LOAD,lfn. lfn Logical file name	Loads program on lfn into central memory	SCOPE 3.4 RM, 2
MAP,p. p ON, OFF, or PART	Selects map, no map, or partial map option	SCOPE 3.4 RM, 9
REDUCE.	Reduces field length of job after loading	SCOPE 3.4 RM, 4

Card Format	Function	Reference Section
PROGRAM EXECUTION CARDS		
lfn,p1,p2,...pn. lfn Name of file containing program p Parameters passed to program	Loads and executes program	SCOPE 3.4 RM, 2
EXECUTE,lfn,p1,p2,...pn. lfn Name of file containing program p Parameters passed to program	Completes loading and linking of elements for execution, then executes this program	SCOPE 3.4 RM, 2
PROGRAMMING OPTION CARDS		
CKP.	Establishes checkpoint	SCOPE 3.4 RM, 10
COMMENT.n...n n Comment characters	Inserts comments	SCOPE 3.4 RM, 4
EXIT.	Establishes exit path in event of selected errors	SCOPE 3.4 RM, 4
EXIT(S)	Establishes exit path after control card or assembly errors	SCOPE 3.4 RM, 4

Card Format		Function	Reference Section
MODE,n.		Defines halt conditions	SCOPE 3.4 RM, 4
n	Type of halt		
RESTART,lfn,n,S=xxx.		Restarts job at checkpoint	SCOPE 3.4 RM, 10
lfn	Name of checkpoint file		
n	Decimal number of checkpoint where job is to be restarted		
xxx	Decimal number of words in smallest physical record (used only with S tape)		
SWITCH,n.		Sets switch to on or off	SCOPE 3.4 RM, 4
n	Sense switch number		

PERMANENT FILE CARDS (See chart, page 5-8, for parameter requirements and options)

Card Format	Function	Reference Section
ALTER,lfn. lfn Logical file name	Sets EOI to current position	SCOPE 3.4 RM, 5
ATTACH,lfn,pfn,ID=name,PW=list,CY=cy or LC=lc,PP=priv,MR=mr,PS=ps,RW=rw,EC=ec. lfn Logical file name pfn Permanent file name name Creator identifier list Passwords cy Cycle number lc Lowest cycle priv Privacy procedure mr Multi-read access ps Attach at SETP position rw Multi-read with single rewrite or single extend access ec ECS buffer size	Assigns permanent file to job	SCOPE 3.4 RM, 5

Card Format		Function	Reference Section

CATALOG,lfn,pfn,ID=name,PW=list,
RP=ret,CY=cy,TK=tk,RD=rd,EX=ex,
MD=md,CN=cn,PP=priv,AC=ac,RW=rw,
FO=fo,XR=xr,MR=mr.
		Makes mass storage file permanent	SCOPE 3.4, 5

lfn	Logical file name		
pfn	Permanent file name		
name	Creator identifier		
list	Password list		
ret	Retention period		
cy	Cycle number		
tk	Turnkey password		
rd	Read password		
ex	Extend password		
md	Modify password		
cn	Control password		
priv	Privacy procedure		
ac	Account parameter		
rw	Multi-read with single rewrite or single extend access		
fo	File organization IS or DA		
xr	Password for modify, extend, and control permissions		
mr	Multi-read access		
EXTEND,lfn.		Makes permanent extension to permanent file	SCOPE 3.4 RM, 5
lfn	Logical file name		

Card Format	Function	Reference Section
PURGE,lfn,pfn,ID=name,PW=list, {CY=cy,} PP=priv,EC=ec,PS=ps. {LC=lc}	Drops permanent file from system	SCOPE 3.4 RM, 5

lfn	Logical file name
pfn	Permanent file name
name	Creator identifier
list	Password list
cy	Cycle number
lc	Lowest cycle
priv	Privacy procedure
ec	ECS buffer size
ps	Attach at SETP position

Card Format	Function	Reference Section

RENAME,lfn,pfn,ID=name,RP=ret,
CY=cy,TK=tk,RD=rd,EX=ex,MD=md,CN=cn,
AC=ac,PW=list,XR=xr.
	Renames control information for permanent file	SCOPE 3.4 RM, 5

lfn	Logical file name
pfn	Permanent file name
name	Creator identifier
ret	Retention period
cy	Cycle number
tk	Turnkey password
rd	Read password
ex	Extend password
md	Modify password
cn	Control password
ac	Account parameter
list	Password list
xr	Password for control, modify and extend permission

Card Format	Function	Reference Section
SETP,lfn. lfn Logical file name	Stores current file position in PF table	SCOPE 3.4 RM, 5
FILE/RECORD MANIPULATION CARDS		
BKSP,lfn,n. lfn Logical file name n Number of records (decimal) to be backspaced	Backspaces n SCOPE logical records	SCOPE 3.4 RM, 4
REWIND,lfn1,lfn2,...lfnn. lfn Logical file name	Rewinds files named	SCOPE 3.4 RM, 4
SKIPB,lfn,n,lev,m. lfn Logical file name n Number of records (decimal) to skip lev Level number of SCOPE logical records to skip m B for binary files; C for coded files	Skips backward by n SCOPE logical records	SCOPE 3.4 RM, 4
SKIPF,lfn,n,lev,m. lfn Logical file name n Number of records (decimal) at or greater than lev to skip lev Level number of SCOPE logical records to be skipped m B for binary records; C for coded records	Skips file forward by n SCOPE logical records	SCOPE 3.4 RM, 4

Card Format	Function	Reference Section
COPYING, COMBINING, AND COMPARING CARDS		
COMBINE,lfn1,lfn2,n. lfn1 Input file lfn2 Output file n Number of records (decimal)	Combines input file records into one logical record of level 0 on output file	SCOPE 3.4 RM, 8
COMPARE,lfn1,lfn2,n,l,e,r. lfn Logical file name n Number of records (decimal) to compare l Level number of SCOPE logical records e Number of non-comparable words to be written r Number of records to be processed during comparison	Compares pairs of files or records	SCOPE 3.4 RM, 8
COPY,lfn1,lfn2. lfn Logical file name	Copies all files in a volume lfn1 to lfn2	SCOPE 3.4 RM, 8
COPYBF,lfn1,lfn2,n. lfn Logical file name n Number of files (decimal)	Copies n binary files from lfn1 to lfn2	SCOPE 3.4 RM, 8
COPYBR,lfn1,lfn2,n. lfn Logical file name n Number of records (decimal)	Copies n binary records from lfn1 to lfn2	SCOPE 3.4 RM, 8

Card Format	Function	Reference Section
COPYCF,lfn1,lfn2,n. lfn Logical file name n Number of files (decimal)	Copies n Hollerith or External BCD files from lfn1 to lfn2	SCOPE 3.4 RM, 8
COPYCR,lfn1,lfn2,n. lfn Logical file name n Number of records (decimal)	Copies n Hollerith or External BCD records from lfn1 to lfn2	SCOPE 3.4 RM, 8
COPYBCD,lfn1,lfn2,n. lfn Logical file name n Number of records (decimal)	Copies packed output files to tape for subsequent off-line listing	SCOPE 3.4 RM, 8
COPYL,lfn1,lfn2,lfn3,program. lfn1 Old set of records lfn2 New set of records lfn3 Updated set program Last record on lfn 1 to be copied	Replaces routines in a file	SCOPE 3.4 RM, 8
COPYN,p,out,in1,in2,...in10. p Record format out Output file in Input file	Input from up to 10 binary files copied to output file (Input directive record required)	SCOPE 3.4 RM, 8
COPYSBF,lfn1,lfn2. lfn1 Input file lfn2 Output file	Copies lfn1 to lfn2 formatting binary file for single space printing	SCOPE 3.4 RM, 8

Card Format	Function	Reference Section
COPYXS,xlfn,scplfn,n. xlfn Input X tape must be requested in S format scplfn Output tape must be requested in S format n Number of files	Converts binary X tapes to S tape format	SCOPE 3.4 RM, 8

Card Format	Function	Reference Section

TAPE PROCESSING CARDS

LABEL,lfn,$\left\{\begin{matrix}R,\\W\end{matrix}\right\}\left\{\begin{matrix}Y,\\Z\end{matrix}\right\}$L=fnl,V=reel,E=ed,
T=ret,C=create,F=df,M=mfn,P=pos,
D=den,N=char,X=dc,VSN=vsn.
	Writes or checks labels on tape file	SCOPE 3.4 RM, 4

lfn	Logical file name
R	Read file
W	Write file
Y	3000 series label
Z	SCOPE 3.3 standard label
fnl	File label name
reel	Reel number
ed	Edition number
ret	Retention period
create	Creation date
df	Data format
mfn	Multi-file name
pos	Number of referenced file in a multi-file volume
den	Density of data on tape
char	Character set for coded data conversion on 9-track tape
dc	Disposition code
vsn	Volume serial number

Card Format	Function	Reference Section
LISTMF,M=mfn,P=p. mfn Multi-file name p File position number to begin listing	Lists contents of labeled multi-file tape	SCOPE 3.4 RM, 8
VSN,lfn1=vsn1,lfn2=vsn2,.... lfn Logical file name vsn Volume serial number associated with lfn	Equates vsn to file name	SCOPE 3.4 RM, 4
FILE EDITING CARD		
UPDATE,ident=lfn1,...,identn=lfnn, list. ident Type of file lfn Logical file name list Optional parameters specify update modes, comments, rewind, format, etc.	Calls UPDATE to perform program library maintenance (directive record required)	UPDATE RM, 10

Card Format		Function	Reference Section
DUMP CARDS			
DMP,x,y.		Dumps specified area of central memory	SCOPE 3.4 RM, 9
x	Beginning address in dump		
y	Last address in dump		
DMPECS,x,y,f,lfn.		Dumps specified area of ECS	SCOPE 3.4 RM, 9
x	Beginning dump address		
y	Last dump address		
f	Print format		
lfn	File to receive dump		
USER LIBRARY CREATION CARD			
EDITLIB,USER,I=dir,L=list.		Creates USER library file (Directive record required)	SCOPE 3.4 RM, 7
dir	Directive input file		
list	List file of output		

DIAGNOSTIC MESSAGES D

Messages that the SCOPE 3.4 Operating System prints are listed below.

Messages are listed alphabetically on the first four characters only. As a result, some may not be in strict alphabetic order, but they will appear within the same group. Items in which the first characters will change according to the parameters of the job in progress are last. Items beginning with numbers follow the alphabetic lists. The routine that produces each message is listed on the right margin.

Abbreviations that commonly appear in this section:

CH	Channel
CM	Central memory
CMR	Central memory resident
CP	Central processor unit, or card punch
ECS	Extended core storage
EOF	End-of-file
EOI	End-of-information
EOR	End-of-record
EQ	Equipment
EST	Equipment status table
FET	File environment table
FL	Field length
FNT/FST	File name/status table
MLRS	Maximum logical record size
MT	Magnetic tape, 7-track
NT	Magnetic tape, 9-track
PF	Permanent file
PFD	Permanent file directory
PP	Peripheral processor unit
PRU	Physical record unit
RBR	Record block reservation table
RBT	Record block table
RBTC	Record block table catalog
UBC	Unused bit count

A CONDITIONAL TABLE IS KEYED TO PROGRAM BLOCK OR BLANK COMMON. NOT USED. LOADO

A DOUBLE EOF WAS FOUND A / COPYN

If a double EOF appears before a zero-length record when P2 is a /, this message is printed, and all records up to the double EOF are written on the output file.

A NUMERIC EXTENDS BEYOND AN END OF FILE COPYN

P2 specifies more records than exist on the file. COPYN writes all the records, one EOF, and rewinds the file.

A PARAMETER BEGINS BEYOND AN EOF-EOF COPYN

P1 is numeric and causes a skip to the double EOF before P1 is zero.

A PARAMETER IS GREATER THAN 7 CHARACTERS COPYN

ABOVE FILE NULL, DELETED XXXRESQ

EOF was encountered on first read of file.

ABS INPUT NOT (0,0) LEVEL OVERLAY LOAD

ABS LOAD ADR LT RA+100 LOAD

ABSOLUTE INPUT IN USER CALL LOAD

Absolute input not allowed.

ABSOLUTE INPUT IN RELOCATABLE LOAD LOAD

The two types of loads are mutually exclusive.

ACE CALLED WITHOUT RECALL ACE

Issued if recall bit of PP input register is not on when ACE is called.

ACE FUNCTION ADDRESS NOT IN FL ACE

Address specified in CM not in user field length.

ADDFILE FIRST CARD MUST BE DECK OR COMDECK UPDATE

Incorrect first card on file processed by ADDFILE directive. Fatal error.

ADDFILE CARD INVALID ON REMOTE FILE	UPDATE
ADDFILE directive cannot be used for a file to be entered onto OLDPL with a *READ directive. Fatal error.	
ADDITIONAL FINAL RIGHT PAREN(S) HAD TO BE ASSUMED	SEGBILD
ALL YANK, SELYANK, YANKDECK AND CALL CARDS AFFECTED HAVE BEEN CHANGED	UPDATE
ALTER NEEDS EXCL. ACCESS	PFE
AN ID (P1) IS REQUIRED ON ALL TEXT CARDS	COPYN
AT PARAM REQUIRED ON ABOVE FRAME DIRECTIVE	TRAP
AT ADDR. nnnn IN SEGMENT x...x 2 CONFLICTING SEGMENTS CALLED BY SAME WORD	SEGBILD
ATTEMPT TO LOAD MORE THAN ONE PROGRAM ON ABS LOAD	LOAD
ATTEMPT TO LOAD SUPPRESSED BINARY	LOAD
Tried to load program with assembly/compilation errors.	LOAD
AUTO-RECALL ERROR	1EJ
Job terminated because completion bit was already set when job went into auto-recall.	
AUTO-RECALL FLAG NOT SET	MEM
AWAITING STORAGE	1TD
BAD COMPARE	COMPARE
Displayed on console, system and job dayfile if discrepancy occurs during COMPARE. If fatal, operator may drop job.	
BAD LABEL	1DA
Unrecognizable label on disk pack read in response to RPACK control card or a DEVADD type-in, or label with wrong name read in response to RPACK control card. Job terminates from DEVADD type-in error; otherwise system waits for operator to assign another disk pack drive, assign same drive with different pack, or drop job.	

BAD PACK NAME IN REQUEST 5DA,1DA

Name on RPACK or REQUEST control card has bad format; or REMOVE card file name on private pack differs from parameter. Job terminates.

BAD REQUEST NO. IN USER CALL LOAD

Request number exceeds maximum.

BAD LOADER DIRECTIVE LOAD

Card image is shown.

BAD LFN ON NOGO - (name) LOAD

BAD SUBTABLE IN LDSET TABLE LOADO

BAD OCTAL DIGIT 1AJ

All parameters on DMP card must be octal. An octal field in the previous control card is in error.

BAD LABEL-CHECKSUM ERR/EST xx 3PK

3PK checksum error occurred during read of sequential pack label assigned by EST xx; job terminated.

BAD POSITION - IGNORED PFA

Request to attach positioned file was ignored because the position stored by SETP was invalidated as a result of ALTER function.

BEGIN EVICTD,Daa 1MH

Start evicting all dependency group aa and release mass storage.

BINARY RECORD MISSING FROM INPUT COPYN

Logical records requested from system INPUT file must begin with the next logical record on INPUT file.

BLANK TAPE READ 6WM

Return dependent on EP bit setting. No error code returned. No data was received after a 1 second delay. 6WM is loaded to issue this message for 1RT, 1RS, 1NR, and 1R9.

BLANK COMMON TRUNCATED LOAD

Insufficient FL for all of blank common. Also in user calls, when something has been loaded above blank common, prevents it from being declared to be larger.

BLANK NAME x...x USED FOR BOTH CM AND ECS.
x...x REPLACES ONE OCCURRENCE. SEGBILD

BOTH NAMES INDICATING RANGE CANNOT BE * EDITLIB

BUFFER ARGUMENT ERROR 6WM
FILE NAME xxxxxxx
FET ADDRESS xxxxxx

Return dependent on EP bit setting. Error code 22 returned in bits 9-13 of FET code/status. Circular buffer pointers in FET fail to satisfy: 0 ≤ FIRST < LIMIT ≤ FL, FIRST ≤ IN < LIMIT, FIRST ≤ OUT < LIMIT. In the table of disk address those for O.RMR are outside the field length.

BUF3 UNABLE TO HOLD PROGRAM NAME TABLE. ENLARGE BUF3 EDITLIB

BUF3 UNABLE TO HOLD LIBRARY NAME TABLE. ENLARGE BUF3 EDITLIB

BUSY WAIT ON FILE - xxxxxxx DSP

Waiting for file to become inactive.

C= NOT ALLOWED, NO BLANK COMMON AT LOWER LEVEL LOADO

CALLING JOB IS NOT DEPENDENT JDP

Job calling TRANSR, TRANSC, or TRANSF does not have dependency identifier.

CALLING ERROR (CKP) CKP

Bit 0 parameter is not zero, RECALL bit is not set in call to CKP, or parameter list is outside field length. Fatal dayfile message, job terminated.

CANNOT DISPOSE NON ALLOC FILE DSP

DISPOSE function illegal for non-allocatable file. DISPOSE will be ignored.

CANNOT DISPOSE NON-LOCAL FILE DSP

DISPOSE function illegal for non-local file. DISPOSE will be ignored.

CANNOT DISPOSE PERMANENT FILE DSP

DISPOSE function illegal for permanent file. DISPOSE will be ignored.

CANNOT DISPOSE THE INPUT FILE DISPOSE function illegal for INPUT file. DISPOSE will be ignored.	DSP
CANNOT PROCESS ENTRY REQUEST - PARAM AREA OVERWRITTEN To process an ENTRY request, the (user call) parameter area must not be within the loadable area.	LOAD
CANNOT PROCESS FILES REQUEST - 1ST RECORD OF zzzzDF TOO BIG This file was built by Record Manager. Loader expects it to be a certain format.	LOAD
CANNOT PROCESS FILES REQUEST - zzzzzDF TOO SHORT This file was built by Record Manager. Loader expects it to be a certain format.	LOAD
CANNOT REQUEST USER AND SYSTEM EDITLIB WITH THE SAME CONTROL CARD	EDITLIB
CANNOT PLANT TRAP IN ECS	TRAPPER
CANT FIND LIBRARY	SEGBILD,LOADO
CANT GET CM FL, JOB RERUN Attempt to initiate job from input failed because 1IB couldn't get control memory storage. Job was rerun.	1IB
CANT RFL ABOVE IP.MFL User specified RFL exceeds installation defined maximum FL.	1AJ
CANT RFL ABOVE JOBCARD FL User RFL exceeds original job card FL.	1AJ
CANT MEM ABOVE JOBCARD CM	MEM
CANT MEM ABOVE IP.MFL	MEM
CARD NUMBER ZERO OR INVALID CHARACTER IN NUMERIC FIELD Sequence number field on a correction control card is in error. Fatal error.	UPDATE
CEM ADDRESS ERROR System error; routine CEM called to issue file error message, but with incorrect FET address.	CEM
CENTRAL TO REMOTE REQUIRES ID	DSP

CIO CODE NOT DEFINED ON DEVICE Function code in FET is not applicable to device containing file. Return dependent on EP bit setting; error code 30.	CIO
CIO IS NOT A MEMBER OF THE PP LIBRARY	EDITLIB
CKP TAPE MUST BE SCOPE STANDARD FORMAT Cannot be S tape or 9-track tape.	CKP
CKP FROM INTERCOM TERM NOT ALLOWED	CKP
CKP REQUESTED CKP begins processing.	CKP
CKP FILE UNKNOWN (RST) No FNT for checkpoint file.	RST
CKP TAPE INVALID (RESTART) Bad records on CKP tape.	RESTART
CKP PARAM INVALID, GO OR DROP	CHEKPT
COMMON BLOCK REDEFINITION - (name) Subsequent declaration specifies larger size than earlier declaration of labeled common; smaller size used.	LOAD
COMMON FILES NOT SUPPORTED UNDER SCOPE 3.4 Attmept to use COMMON or RELEASE control card.	1AJ
COMPLETE EVICTD,Daa Issued when all of dependency group aa has been evicted.	1MH
COMPLETE REQUIRED BEFORE ANOTHER READY	EDITLIB
COMPLETE DIRECTIVE WAS NOT FOUND BEFORE INPUT WAS EXHAUSTED	EDITLIB
COMPLETE ENCOUNTERED BEFORE A FINISH	EDITLIB
CONSULTING LIBRARY x...x	SEGBILD
CONTAINS ONLY A 77- AND/OR 70- TABLE	SEGBILD
CONTROL CARD ERROR, NO CKPFILE (RESTART) RESTART card has two numbers instead of a number and file name. Job terminated.	RESTART

CONTROL CARD ERROR, NUMBER ERR (RESTART) Checkpoint number must be unsigned decimal integer greater than zero. Job terminated.	RESTART
CONTROL CARD ERROR, S PARAMETER (RESTART) S parameter on RESTART card must begin with the character S. Job terminated.	RESTART
CONTROL CARD REWIND (INPUT) IS ILLEGAL System INPUT file cannot be rewound.	COPYN
CONTROL CARD ERROR. FILE COUNT NON-NUMERIC Illegal character encountered in a field that should contain a number. Job terminated.	COPYXS
CONTROL CARD INVALID OR MISSING Control card has format error or could not be read. Illegal operation such as *INSERT prior to *IDENT may have been attempted. Fatal error.	UPDATE
CONTROL CARD ERROR Control card exceeds 7 characters; illegal characters; or too many parameters. Job terminated.	1AJ
CONTROL CARD ERROR (LABEL) Error in LABEL control card. Job terminated.	LABEL
CONTROL CARD ERROR - ACE Illegal control card.	ACE
CONTROL CARD ERROR lfn exceeds 7 characters; VSN parameter exceeds 6 characters.	VSN
COPYL DID NOT FIND xxxxxxx.	COPYL
COPYL DONE	COPYL
COPYL - ILLEGAL DEVICE TYPE	COPYL
COPY TO EXTERNAL FILE NOT ALLOWED WHEN READING FROM ALTERNATE INPUT UNIT	UPDATE
COPYING OLDPL TO RANDOM FILE	UPDATE
COULDNT FIND PROGRAMS	SEGBILD

COULD NOT FIND BLOCK xxxxxxx TRAPPER

CP PROGRAMS MAY NOT RESIDE IN THE PP LIBRARY EDITLIB

CY NO. LIMIT REACHED PFC

Highest current cycle number is 999.

CY NO. ALREADY IN USE PFC

CYCLE INC. OR DUMPED PFA

Permanent file table information for file is incomplete; file not accessible.

D.FNT+9=xxxx 1NR,1RS

Cell D.FNT+9 contains xxxx. Issued only when IP.DBUG=1.

D.FNT+8=xxxx 1CS

Value of cell D.FNT+8 is xxxx. Issued only when IP.DBUG=1.

DATA EXCEEDS MLRS 1WS,1NW

Fatal error, code 22.

DAYFILE DUMPED 1DF

Dayfile buffer flushed at this point and dump begun.

DECIMAL FIELD CONTAINS A NAME EDITLIB

DECK NAME ON THE ABOVE CARD NOT LAST DECLARED DECK UPDATE

Informative message.

DECK SPECIFIED ON MOVE OR COPY CARD NOT ON OLDPL. CARD WILL BE IGNORED UPDATE

Deck does not exist on OLDPL. Informative message.

DELIMITER MISSING AFTER LITERAL FIELD EDITLIB

DELIMITER FOLLOWS A DELIMITER OR RECORD OR FILE COUNT IS MISSING EDITLIB

DEVICE FULL/FILE MAY NOT OVERFLOW CEM,6WM

DEVICE TYPE CONFLICT. SHOULD xx BE ASSIGNED REQ

Device type and EST ordinal differ in user REQUEST.

DEVICE CAPACITY EXCEEDED COPYCF,COPYCR, COPYBR

DIRECTIVE OR UNRECOGNIZABLE INPUT IN ABS LOAD LOAD

Directives are not allowed on absolute loads.

DIRECTIVE FRAGMENT LOAD

EOR detected before end of directive reached, e.g., OVERLAY(FN followed by EQR status).

DIRECTIVE LONGER THAN 80 CHARS LOAD

No zero byte terminator after 80 characters.

DISP CODE NEEDED IF ID GIVEN DSP

Disposition code must be present if an identifier is specified in a DISPOSE call; function is ignored.

DISP ID GT 2 CHARS DISPOSE

Identifier on a DISPOSE card must be 1 or 2 characters. DISPOSE card ignored.

DISPOSE ARGS NOT IN FL DSP

DISPOSE function will be ignored.

DISPOSE IGNORED, NO FILE -- xxxxxx DSP

File to be disposed could not be found.

DMP - ARG OUTSIDE FL DMP

The beginning, or x parameter, is not within job field length. No dump will result.

DMP - FWA EXCEEDS LWA DMP

Beginning parameter exceeds ending parameter. No dump will result.

DMP - LWA EXCEEDS FL - FL SUBSTITUTED DMP

Ending parameter (last-word parameter) is greater than job FL. Field length is substituted and dump is made.

DMP - NO ABSOLUTE DUMPS UNLESS IP.DEBUG=1 DMP

For privacy reasons, installation parameter IP.DEBUG has been turned off (=0). No absolute dumps will be made, but relative dumps (within job field length) will be made as requested.

DMP-FL=ZERO DMP

DP xx END OF PACK — 1PK

Dayfile message when disk pack overflows or premature CLOSER function is issued on a pack with EST ordinal xx.

DROP — EDITLIB

Dayfile message issued if operator responded DROP to an EDITLIB GO/DROP message.

DSD WAS TRANSFERRED BEFORE INTERVAL OR COUNT SATISFIED — EDITLIB

DUMP QUEUE ABANDONED — XXXDMPQ

Issued when FNT is full and DMPQ cannot get an FNT entry for its tape.

DUPLICATE FILE NAME (RESTART) — RESTART

Requested file is already assigned to this control point. Job terminates.

DUPLICATE PACK NAME — 1DA

Pack name on RPACK control card same as a pack name already at that control point. Job terminated.

DUPLICATE DECK NAME ON CREATION RUN — UPDATE

Fatal error.

DUPLICATE IDENT NAME — UPDATE

During a merge run, a duplicate IDENT name was encountered which UPDATE could not make unique. Fatal error.

DUPLICATE KEYWORD — PFCCP

DUPLICATE IDENT NAME IN ADDFILE — UPDATE

In ADDFILE directive, name in file to be added duplicates existing IDENT on OLDPL. Fatal error.

DUPLICATE ENTRY POINT NAME - (name) — LOAD

Same entry point name encountered more than once; first definition holds.

DUPLICATE PROGRAM NAME FROM FILE — LOAD

Same program name encountered more than once; only first gets loaded.

DUPLICATE PROGRAM NAME FROM LIBRARY — LOAD

All programs are loaded.

DUPLICATE FILE NAME - xxxxxxx. — VSN

File name xxxxxx already exists, duplicate will be ignored.

DUPLICATE FILE NAME — 5DA,REQ

Request card asks for new file fname, but it is already at the control point.

DUPLICATE FILE NAME — 1DA

Sent to both dayfiles if existing private pack family is assigned to a control point in response to RPACK control card, and file in family has the same name as the file already at the control point; job terminated.

ECS PARITY ERROR at xx — 1EJ

ECS parity error during a system storage move terminated job. Job may be active at a control point or in job initialization phase requesting storage at a control point. xx is absolute ECS address/100 (octal).

ECS INDEX ERROR — CEM,6WM

Read attempt on an ECS resident random file specified an invalid index.

ECS READ ERROR — LOAD

Lower exit taken on RE instruction.

ECS WRITE ERROR — LOAD

Lower exit taken on WE instruction.

ECS READ ERROR (CKP) — CKP

ECS WRITE ERROR (RESTART) — RESTART

ECS ERROR ON SWAP FILE — CEM

Parity error occurred on swap file in ECS during swap-in, job terminated.

ECS ERROR NUMBER BAD	CEM
System error, routine CEM called with cut-of-bounds error code.	
ECS ERROR LOADING xxxxxxx	CEM
Routine xxxxxxx was called from ECS resident library, and parity error occurred.	
EDITLIB WAS UNSUCCESSFUL IN ITS ATTEMPT TO ATTACH x...x FILE	EDITLIB
EMPTY LOAD	LOAD
Nothing was loaded.	
EMPTY LOAD FILE - (file name)	LOAD
EMPTY LIBRARY PROGRAM READ	LOAD
System error - EOR or EOF status received immediately after read was issued.	
EMPTY FILE	SEGBILD
EMPTY OVERLAY	LOADO
EMPTY RECORD IN LIBRARY	LOADO
EMPTY ON PASS 2	SEGBILD
EMPTY PRESET DIRECTIVE	SEGBILD,LOADO
EMPTY TABLE FOUND	SEGBILD
ENDRUN ENCOUNTERED BEFORE A COMPLETE	EDITLIB
ENDRUN ENCOUNTERED BEFORE A FINISH	EDITLIB
ENTRY ON LIBLOAD NOT FOUND - (name)	LOAD
ENTRY NAME ON NOGO NOT FOUND - (name)	LOAD
Entry points do not exist in the load.	
ENTRY POINT IN BLANK COMMON OR NEG. RELOC.	LOADO
ENTRY POINT COLLECTION TABLE OVERFLOW	EDITLIB
Enlarge program parameter BUF4S and install new version of EDITLIB.	
EOF/EOI ENCOUNTERED BEFORE ALL THE RECORDS WERE TRANSFERRED	EDITLIB

EOI/EOR/EOF ENCNTRD PREMATURELY RESTART

EOI/EOR/EOF encountered before the checkpoint tape is repositioned correctly.

EOJ DISPOSITION SPECIFIED, ky IGNORED DSP

ky identifier for remote terminal or equipment characteristic can be handled only with intermediate DISPOSE.

EOF BEFORE x...x FOUND EDITLIB

EOF was encountered before the interval was satisfied.

EOR/EOF ENCOUNTERED BEFORE LITERAL FIELD TERMINATOR FOUND EDITLIB

EOR/EOF ENCOUNTERED WHILE IDENTIFYING A DIRECTIVE EDITLIB

EPNT LIST OVERFLOW - INCREASE THE SIZE OF BUF4 EDITLIB

Enlarge program parameter BUF4S and install new version of EDITLIB.

EPNT OVERFLOW WHILE ADDING PGM x...x TO LIBRARY EDITLIB

EPNT OVERFLOW EDITLIB

More than 2047 entries in entry point name table.

EQUIPMENT NOT AVAILABLE - DROP OR RECHECK RESTART

Status returned from REQ.

ERROR nn ON OVERWRITE CALL TO LDV LDW

The overlay loading routines LDV and LDW normally issue no error messages but return an error code in the reply word of the calling sequence. However, if a call specifies that the calling program may be overwritten by an overlay load, no attempt can be made to update the calling sequence. Instead, the above message is issued, and the job is terminated. nn is one of the following standard LDV-LDW error conditions.

1 Specified LFN does not exist (non-library loads).
2 Specified library name does not exist.
3 Overlay not found.
4 Insufficient FL for overlay.
5 Overlay header of incorrect format.
6 Format error in request.
7 Both auto-recall and E=1 specified.
10 Index error on user library.
11 I/O error status returned by system.

ERROR CONDITION NOT CLEARED LAST REQUEST   CIO,6WM

Fatal error. Error code 22. FET shows that an error occurred on the previous function. Program did not clear error status.

ERROR IN PARAD (1RC)   1RC

Address of parameter list is outside field length. Fatal dayfile message; job terminated.

ERROR MODE = x, ADDRESS = xxxxxx   1EJ

Program terminated because of address or operand error. If x = 0, program attempted to jump to location 0. See error mode numbers under MODE control card.

ERRORNOT ALL MODS WERE PROCESSED   UPDATE

All changes indicated in the input deck were not processed. Fatal error. Names specified on correction cards should correspond to deck or identifier names existing on OLDPL.

ERROR DUPLICATE ENTRY POINT NAME IN PGM NAME x...x   EDITLIB

ERROR, DIRECTIVE NUMBER xx   EDITLIB

A message relative to the error detected will accompany this message.

ERROR IN FL (1RC)   1RC

New FL must be at least 300 (octal) and greater than old FL.

ERROR AT EOF0, NOTIFY SYSTEMS   XXXDMPQ

EOI was encountered on first read of a file; cannot logically occur.

ERROR IN DISPOSE FUNCTION   DSP

Identifier must be 2 alphanumeric characters.

ERRORS IN ONLY GEN   LOADO

ERT EXCEEDS MAX. TABLE SIZE. DECREASE THE NO. OF PGM IN THIS LIBRARY   EDITLIB

EST CONFLICT. SHOULD EST xx BE ASSIGNED   REQ

EST ordinal assigned by operator differs from user REQUEST.

EST IS WRONG DEVICE TYPE. SHOULD xx BE ASSIGNED   REQ

Device type assigned by operator differs from user REQUEST.

EST ORDINAL TOO HIGH 1DA

A BLANK, DEVADD, UNLOAD, or PACK type-in gives EST ordinal that exceeds length of EST; job terminated.

ESTO uu VSNO 0000visum 1DA

Sent to both dayfiles and the B display in response to type-in of n.PACK,E=uu,V=visum.

ESTO uu 1DA

Sent to both dayfile and the B display in response to type-in of n.PACK,E=uu.

EVICT ILLEGAL ON PERMANENT FILE CIO

Return dependent upon setting of EP bit. Error code 32.

EVICT ILLEGAL ON SEQUENTIAL PACK FILE 1PK

Return dependent upon setting of EP bit. Error code 24.

EXECUTE OR NOGO ARE ERROR LOAD

Format error detected on either EXECUTE or NOGO control card.

EXTEND FUNCTION UNSUCCESSFUL ON SYSTEM EXTENSION FILE EDITLIB

EXPORT/IMPORT NOW PROCESSED BY INTERCOM

EXTERNAL REFERENCE COLLECTION TABLE OVERFLOW EDITLIB

Enlarge program parameter BUF5S and install new version of EDITLIB.

EXTERNAL x...x MISSING ON PASS 2 SEGBILD

FAMILY 0000000, VRNO 000000,ESTD 00 EKG

FATAL ECS PARITY ERROR - TYPE GO 6SI

Fatal parity error detected when job was swapped-in from ECS.

FATAL ERROR -- LAST PROG. READ WAS x...x FROM FILE xxxx SEGBILD,LOADO

FATAL LOADER ERROR LOADO,SEGBILD

When fatal error is encountered dayfile message is followed by a message stating probable cause.

FDB ADDR ERROR PFM

FDB address is outside job's field length.

FET OUTSIDE FL CIO,6WM

All or part of file environment table outside user field length. Fatal error, code 22.

FET TOO SHORT 1RS,1WS,1RT,1R9,1NR,1NW, CIO,1W9,1MT

Fatal error, code 22.

FIELD IS NON NUMERIC ILLEGAL TEXT CARD COPYN

FIELD LENGTH OVERRIDE BIT CANNOT BE SET BY THE FL PARAMETER EDITLIB

FILE MAY NOT RESIDE ON DEVICE ASSIGNED 3DO,2TB,1NR,1RS,1R9,1RT, 1WS,1NW,1W9,1WI,1MT

Fatal error, code 22. Operator assigned incorrect device.

FILE NAME NOT SPECIFIED DISPOSE

File name must be specified on a DISPOSE card. DISPOSE card will be ignored.

FILE NAME GT 7 CHARS DISPOSE

File name on DISPOSE card must not exceed 7 characters; card will be ignored.

FILE NAME ON UPDATE CARD GT 7 CHAR, UPDATE ABORTED UPDATE

Dayfile message.

FILE DID NOT PROVIDE PROGRAMS SEGBILD

FILE NAME ON ABOVE CARD GREATER THAN SEVEN CHARACTERS UPDATE

Fatal error.

FILE x...x EMPTY LOADO

FILE x...x ON THIS FAMILY WANDERED 1EJ

FILE NAME NOT IN MULTI-FILE SET (LABEL) LABEL

Specified file name not in multi-file set. Job terminated.

FILE EVICTED BY THE OPERATOR - jobname EVICT,lfn 1MH

Remove lfn from the input, output, and punch queues. Mass storage is released.

FILE NOT IN TAPE QUEUE 1MH

Illegal name as tape job not in queue.

FILE LIMIT EXCEEDED - USE LESS EDITLIB

If number of files cannot be decreased, increase program parameter MFILES and install new EDITLIB version.

FILE NAME MUST BEGIN WITH AN ALPHABETIC CHARACTER EDITLIB

FILE NAME CONTAINS AN INCOMPLETE BINARY RECORD EDITLIB

FILE x...x IS NOT A LIBRARY EDITLIB

FILE NAME TOO BIG (RESTART) RESTART

Refers to checkpoint file.

FILE NOT ON PRIVATE PACK 5DA

File named on REMOVE card exists, but does not reside on a family pack.

FILE NEVER ASSIGNED TO A DEVICE PFC

FILE ALREADY IN SYSTEM LPF

Attempt to load a file which is already in system.

FILE NOT IN SYSTEM PFA

File identified by PFN, ID, and cycle number does not exist.

FILE NOT ON PF DEVICE PFC

File does not reside entirely on mass storage devices designated by installation for permanent files.

FILE UNAVAILABLE PFA

RT parameter specified in system macro call, but file not available because: Job requesting file wants exclusive access or other jobs already have exclusive access; attached permanent file table is full; archived file temporarily unavailable; permanent file utility routine is running.

FINISH REQUIRED BEFORE ANOTHER LIBRARY EDITLIB

FINISH AND COMPLETE REQUIRED BEFORE ANOTHER READY EDITLIB

FINISH REQUIRED BEFORE A READY EDITLIB

FIR=nnnnnnnn LI=xxxxxxxx 1CR

Value of FIRST is nnnnnnnn. Value of LIMIT is xxxxxxxx. Issued only if IP.DBUG=1.

FIRST DIRECTIVE NOT ((LOAD)) LOADO

FIRST OVERLAY DIRECTIVE NON-ZERO LEVEL LOADO

FIRST OVERLAY DIRECTIVE HAS NO FILE NAME LOADO

FIRST CHARACTER MUST BE ALPHABETIC IN A LIBRARY NAME EDITLIB

FL INCREASED BY 2000 WORDS TO ACCOMMODATE TABLES EDITLIB

FLO VALUE CAN ONLY BE 0 OR 1. FLO VALUE WILL REMAIN UNCHANGED EDITLIB

FNT FULL 1EJ

Output file cannot be created as FNT is full. Loop will continue until empty entry is found.

FNT FULL VSN

No FNT space available.

FNT NAME ERROR (1RC) 1RC

Refers to memory dump file.

FNT FULL - DROP OR RECHECK RESTART

FO PARAMETER INVALID - IGNORED PFC

Logical file name does not reference IS or DA file.

FOLLOWING PF CONTROL CARD IN ERROR PFCCP

Permanent file control card contains format error. The following reasons apply:

Could not match function name in list of permanent file functions.

Permanent file manager function name not followed by blank, comma, or left parenthesis.

Local file name not followed by comma or terminator.

Keyword followed by other than equals sign.

Keyword which demands a numeric parameter has an alphabetic parameter.

Control card not properly terminated.

Permanent file utility function name not followed by blank, comma, left parenthesis or terminator.

Illegal use of dollar sign.

Comma or terminator does not follow a delimited string.

FORMATS INCOMPATIBLE***COPY UNCERTAIN*** COPYCR,COPYCF

Tape formats selected for input/output files can cause a possible loss of data significance. Job continues.

FORMAT ERROR ON LIBLOAD REQUEST LOAD

FORMAT ERROR ON CMLOAD OR ECLOAD REQUEST LOAD

FORMAT ERROR - SATISFY REQUEST - (name) LOAD

FORMAT ERROR ON LIB REQUEST - (name) LOAD

FORMAT ERROR ON MAP REQUEST LOAD

FORMAT ERROR ON PRESET REQUEST	LOAD
FORMAT ERROR ON USEP REQUEST - (name)	LOAD
FORMAT ERROR ON USE REQUEST - (name)	LOAD
FORMAT ERROR ON OMIT REQUEST - (name)	LOAD
FORMAT ERROR IN ABOVE DIRECTIVE	LOADO
FUNCTION ATTEMPT ON NON-LOCAL FILE	PFC
Logical file name specified or assumed is an attached permanent file.	
FUNCTION ATTEMPT ON NON-PERMANENT FILE	PFM
FWA-LWA ERROR	IO,COMPARE
Console, system and job dayfile. READIN or WRITOUT macro is executing; workspace (sixth word of FET) non-existent or FET is less than 6 words. Job aborted.	
GARBAGE IN OLDPL HEADER, UPDATE ABORTED	UPDATE
GOOD COMPARE	COMPARE
GOOD MORNING	IGM
Displayed by IGM at the start of a new day.	
HAS 0 LINK ADDRESS. PROCESSING UNCERTAIN	XDQ
LINKED FNT entry has zero link address.	
I/O ERROR (CKP)	CKP
Error returned from either REQ or CIO.	
I/O ERROR (1RC)	1RC
CIO returned error.	
I/O ERROR DURING COMBINE	COMBINE
Dayfile will have appropriate information from the drivers.	
ID ERROR	PFC,PFR
The ID of SYSTEM was specified on initial CATALOG or RENAME attempt, or the ID of PUBLIC was specified on initial CATALOG or RENAME attempt without the public permission password.	

ID NAME NOT IN INPUT FILES SEARCHED   COPYN

COPYN routine cannot locate either P1 or P2.

ID-PFN PAIR ALREADY EXISTS, IGNORED   PFR

Owner-ID and permanent file name on a RENAME request match an existing owner-ID and permanent file name. RENAME, in this case, will not change either.

IDENT CARD MISSING, NO NEWPL REQUESTED, DEFAULT IDENTIFIER OF .NO.ID. USED   UPDATE

Informative message.

IDENTIFIERS SEPARATED BY PERIOD IN WRONG ORDER   UPDATE

Identifiers on PURGE, COMPILE, or SEQUENCE cards were in the wrong order. Fatal error.

ILL-FORMED LINK TABLE   EDITLIB

ILL-FORMED MULTIFILE SET   1MF,1TO

Fatal error, code 22. Multi-file set tape not created in standard ANSI format.

ILL-FORMED TABLE OF CONTROL CARDS   SEGBILD,LOADO

ILL-FORMED HOLD FILE   LOADO

Indicates an internal LOADO problem.

ILLEGAL COPYL PARAMETER   COPYL

ILLEGAL DEVICE FOR FILE (filename)   COPYN

ILLEGAL FILE NAME   CIO

File name has embedded blanks, does not begin with letter, or exceeds 7 characters. Fatal error, code 22.

ILLEGAL I/O REQUEST   6WM,CIO, all I/O drivers

First of a 4-line message issued as a result of a call from I/O driver, followed by:
    FILE NAME xxxxxxx
    FET ADDRESS yyyyyy
Fourth line describes error condition.

ILLEGAL LEVEL NUMBER   SKIPF,SKIPB

ILLEGAL PARAMETER LIST JDP,TRANSR

Parameters in TRANSR, TRANSF and TRANSC control cards must be a legal job name left justified, zero filled, and less that 5 characters.

ILLEGAL REQUEST FUNCTION (RESTART) RESTART

Error occurred when RESTART attempted to request tape. Job terminated.

ILLEGAL REQUEST PARAMETER REQ

Request is aborted.

ILLEGAL CONTROL CARD IN ADDFILE UPDATE

ADDFILE insertions cannot contain correction directives. Fatal error.

ILLEGAL FUNCTION CODE CIO

ILLEGAL FILE NAME IN REQ CALL REQ

File name in REQ parameter list is not a legal SCOPE file name; job terminates.

ILLEGAL DEVICE TYPE IN REQUEST REQ

Device type in REQ parameter list is not valid; job terminates.

ILLEGAL LOADER REQUEST LOAD

ILLEGAL EQUIPMENT REQUEST 1DF

Illegal equipment parameters passed from operator type-in.

ILLEGAL JOB NAME JDP

ILLEGAL POSITION NUMBER (LABEL) LABEL

Position number on LABEL card beyond legitimate range. Job terminated.

ILLEGAL TAPE REQUEST (LABEL) LABEL

Local file at this control point is not magnetic tape file. Job terminated.

ILLEGAL PARAMETERS (LISTMF) Unrecognizable parameter on LISTMF card. Job terminated.	LISTMF
ILLEGAL POSITION NUMBER (LISTMF) Invalid position number on LISTMF card. Job terminated.	LISTMF
ILLEGAL ACE FUNCTION Only read and backspace function legal.	ACE
ILLEGAL FILE NAME-OUTPUT OUTPUT cannot be used for lfn. Job terminated.	VSN
ILLEGAL DELIMITER FOLLOWING x...x - REPLACE x...x WITH x...x OR x...x	EDITLIB
ILLEGAL CY1 CALL (CY1/RESTART) Bad flag in CALL to CY1 from RESTART.	CY1
ILLEGAL RESTART PARAMETER Checkpoint file name does not begin with alphabetic or numeric character.	RESTART
ILLEGAL PARAMETER FOR COMBINE Record count non-numeric or illegal file name.	COMBINE
ILLEGAL VERB IN ABOVE TRAP DIRECTIVE	TRAP
ILLEGAL DELIMITER ON ABOVE TRAP DIRECTIVE	TRAP
ILLEGAL PARAM ON ABOVE TRAP DIRECTIVE	TRAP
ILLEGAL NUMBER ON ABOVE TRAP DIRECTIVE	TRAP
ILLEGAL DEVICE First file assigned to device that cannot be read, or second file to device that cannot be written.	COPYCF,COPYCR, COPYBR,COPYBF
ILLEGAL NUMBER OF RECORDS Number of records parameter is not a valid decimal digit.	SKIPB,SKIPF
ILLEGAL CALL TO DUMP QUEUE DMPQ was called by control card, but must be typed in.	XDQ

ILLEGAL KEYWORD
PFCCP

Keyword greater than two characters; keyword not found in list of legal keywords; null keyword.

ILLEGAL LFN
PFA

Logical file name does not begin with alphabetic character.

IMPROPER UPDATE PARAMETER, UPDATE ABORTED
UPDATE

Dayfile message. Unrecognizable parameter on update card.

IN=nnnnnnnn OUT=xxxxxxxx
1CS,1CR

Issued only when IP.DBUG=1.

INACTIVE CYCLE, NO PFN, ID, CY CHANGES MADE
PFR

A rename is not allowed if the cycle referenced has incomplete permanent file table information.

INCLUDE MAY NOT APPEAR BETWEEN A LIBRARY AND FINISH.
EDITLIB

INCLUDEP INVALID IF PPNT IS NOT EMPTY
EDITLIB

INCORRECT PERMISSION
PFM

Control permission not granted for PURGE, addition of a new cycle, or attach of incomplete cycle.

Extend permission not granted for EXTEND, ALTER or SETP.

Modify permission needed for ALTER function.

INCREASE FL AND RERUN JOB
EDITLIB

Issued when a valid run can be obtained by increasing the FL of job.

INDEX ADDRESS NOT IN FL 1OP,1CL,CIO

Fatal error, code 22. All or part of specified index buffer is not within control point field length.

INDEX BUFFER NOT SPECIFIED IN FET 1CL

Non-fatal error. Index buffer not specified on close of random file. No index is written.

INDEX BUFFER TOO SMALL 1OP,CIO

Return dependent on EP bit setting. Error code 23. Specified index buffer is too small to contain the index logical record. First part of record was read.

INDEX MAY BE INVALID - FILE WRITTEN SINCE LAST CLOSE 1CL

OPEN requested on random file which was not closed (but had index written) since last write operation. Non-fatal diagnostic.

INITIAL CATALOG PFC

INPUT AND OUTPUT TAPES NOT BOTH S FORMAT COPYXS

REQUEST card for both input and output tape must specify S format. Job terminated.

INPUT EXHAUSTED BEFORE FINDING A FINISH DIRECTIVE EDITLIB

INPUT EXHAUSTED BEFORE FINDING AN ENDRUN DIRECTIVE - ONE GENERATED EDITLIB

INSUFFICIENT FIELD LENGTH COPYXS

Minimum field length required is 12000 octal. Job terminated.

INSUFFICIENT FIELD LENGTH 1AJ

Program call card initiated load of absolute program or 0,0 overlay, but insufficient field length caused job termination.

INSUFFICIENT FL FOR LOAD LOAD

INSUFFICIENT ECS FL FOR LOAD LOAD

INTERCOM NOT ON THIS SYSTEM DSP

User tried to dispose a file to an INTERCOM terminal, but INTERCOM not implemented on system. DISPOSE function is ignored.

INTERCOM TO CENTRAL, y IGNORED DSP

INTERVAL CANNOT BE SATISFIED - UNABLE TO LOCATE FIRST PGM   EDITLIB

INTERVAL CANNOT BE SATISFIED - UNABLE TO LOCATE LAST PGM   EDITLIB

INTERVAL CANNOT BE SATISFIED - PP PROGRAM LIBRARY IS FULL   EDITLIB

Increase FL and rerun job. If error occurs again, increase program parameter MAXLPPNT and install new version of EDITLIB.

INTERVAL IMPROPERLY FORMED - x...x PRECEDES x...x   EDITLIB

INVALID REQUEST TO CPC   CPC,COMPARE

Bits 54-59 of request word are 0, so it cannot contain PP program name in bits 42-59. Bits 42-59 can contain only number below 00010 (octal) as part of file action macro. Console, system and job dayfiles. Job terminated.

INVALID NUMERIC FIELD   UPDATE

Control card did not contain anticipated numeric information in a numeric field. Fatal error.

INVALID EST ENTRY   REQ

Invalid EST ordinal specified in user request or operator assignment.

INVALID PARAMETER AFTER LEVEL NUMBER   LOADO

INVALID CYCLE NUMBER IGNORED   PFR,PFC

Cycle number specified was either 0 or greater than 999; no number change will be made.

INVALID CONTINUATION CARD (LABEL)   LABEL

Continued control card not found. Job terminated.

INVALID STACK ENTRY   1SX
FILE NAME xxxxxxx
FET ADDR xxxxxx

Request stack entry contains an undefined or invalid order code; a user has not specified an FET for the order code, O.RMR (read several records).

INVALID CONTROL WORD OR FWA(RA+100) — MSG

Special call from 1AJ.

JDP AUTO RECALL ERROR — JDP

JOB HUNG IN AUTO-RECALL — 1EJ

This job terminated because completion bit was not set and control point had no activity.

JOB PRE-ABORTED — 1EJ

Appears at beginning of job output file if input card caused CKSUM ERROR RC.xxxx, CD.yyyy or FORMAT ERROR RC.xxxx, CD.yyyy. Job terminated when brought to control point.

JOB REPRIEVED — RPV

Terminated user job was restarted in the recovery routine.

JOB KILLED — 1EJ

JOB RERUN — 1EJ

JOB CARD ERROR — 1EJ

JOB WAS RERUN — 1IB

JOB SWAPPED IN — 1SI

jobname CONTROL POINT ERROR — 1LT

Error flag set other than operator drop.

jobname CONTROL POINT ABORTED — 1LT

Error code returned in FET.

KEYWORD HAS TOO MANY PARAMETERS — PFCCP

Either PW was equated to more than five parameters, or LP was was equated to more than four parameters.

LA=xxxx WC=yyyy

Value of cell LA is xxxx. Value of cell WC is yyyy. Issued only when IP.DBUG=1.

LAST DEADSTART RECORD ALREADY TRANSFERRED. DIRECTIVE IGNORED — EDITLIB

LAST TABLE UNFINISHED	SEGBILD
LAST TABLE UNFINISHED ON PASS 2	SEGBILD
LATEST INDEX NOT WRITTEN	PFC,PFE
Latest index has not yet been written on random file specified by logical file name.	
LENGTH ERROR ON OLDPL. UNUSABLE OLDPL OR HARDWARE ERROR	UPDATE
Card length on OLDPL exceeded maximum allowed, or it was less than one. Fatal error.	
LENGTH OF x...x EXCEEDS AVAILABLE ECS SPACE	EDITLIB
LFN GREATER THAN 7 CHARACTERS	SKIPF,SKIPB,BKSP
LFN parameter of skip control card is too long.	
LFN EXCEEDS 7 CHARS	PFCCP
LFN IS (logical file name)	PFM
If the logical file name is assumed, the assumed name is printed as part of the message.	
LFN ALREADY IN USE	PFA
Logical file name specified is already in use by this job.	
LIB= IS NOT VALID IN USER EDITLIB MODE	EDITLIB
LIBRARY PROGRAM ENDS BEFORE EOR	LOADO
LIBRARY MUST PRECEED A FINISH	EDITLIB
LIBRARY NAME IS MISSING OR DELIMITER FOLLOWS A DELIMITER	EDITLIB
LIBRARY MUST PRECEED A DELETE	EDITLIB
LIBRARY MUST PRECEED AN ADD OR REPLACE	EDITLIB
LIBRARY NAME IS MISSING OR INVALID DELIMITER WAS USED	EDITLIB
LIBRARY OR READY MUST PRECEED AN ADD OR REPLACE	EDITLIB
LIBRARY OR READY MUST PRECEED A DELETE	EDITLIB
LIBRARY OR READY MUST PRECEED A SETAL	EDITLIB
LIBRARY OR READY MUST PRECEED A SETFL OR SETFLO	EDITLIB

LIBRARY MUST IMMEDIATELY PRECEED OLDLIB DIRECTIVE	EDITLIB
LIBRARY PARAMETER IS NOW NEW	EDITLIB
LIBRARY MUST BE ON A RANDOM FILE IF OLD PARAMETER SPECIFIED	EDITLIB
LIMIT ZERO IGNORED	1AJ
Attempt made to set unlimited mass storage. Not fatal.	
LISTLIB MAY NOT APPEAR BETWEEN A LIBRARY AND FINISH	EDITLIB
LISTLIB IS VALID FOR NEWSYS / RUNSYS BETWEEN READY - COMPLETE	EDITLIB
LOADER CARD ERROR	LOAD
Card image is shown.	
LOAD FILE NAME FORMAT ERROR - (name)	LOAD
LOAD CONTROL POINT IN USE	1LT
1LT assigned to a control point that already has a job running.	
LOAD COMPLETE	1LT
Requested job load is complete.	
LOAD TERMINATED DUE TO EXIT CARD CALLING ACE TO BACKSPACE	LOAD
EXIT control card detected within a sequence of loader control cards. Backspace is performed so that EXIT card can be processed in usual manner.	
LOC ARG ERROR	LOC
Incorrect parameter, such as last word address greater than field length.	
LOCAL FILE CONTAINS AN INCOMPLETE LIBRARY	EDITLIB
LOCAL FILE IS NOT A DS TAPE OR USER LIBRARY	EDITLIB
LOGICAL RECORD ENDS IN MID-TABLE	LOADO
LOGICAL FILE NAME IS MISSING OR INVALID DELIMITER WAS USED	EDITLIB
LOST FILE (CY1/RESTART)	CY1
No FNT for file from RESTART.	
LOST FNT LINK FOR 2MT (CY1/RESTART)	CY1

M PARAMETER MISSING (LISTMF)	LISTMF
Multi-file name not specified on LISTMF card. Job terminated.	
MASS STORAGE LIMIT	1EJ
MAXIMUM TAPE UNITS EXCEEDED	REQ,1TS
Full allotment of tape units already assigned to control point. Job terminated.	
MDF NOT MASS STORAGE (1RC)	1RC
Memory dump file must be disk file.	
MDI - FILE NOT FOUND - xxxxxxx	MDI
EDITLIB attempt to find the named file failed. Job may kill MTR depending on which file is unavailable, or it may continue if file is not vital to system.	
MDI ABORT - BAD DIRECTORY POINTER	MDI
System error. Directory pointers illogical. Job terminated.	
MDI ABORT - BAD PARAMETER	MDI
MDI called with bad parameters. Job terminated.	
MDI ABORT - ERROR FLAG SET	MDI
MDI ABORT - ILLEGAL REQUEST	MDI
Illegal operation request in call by system routine.	
MEM ARG ERROR - STATUS ALREADY COMPLETE	MEM
Request made to MEM when completion bit was set in status word.	
MEM ARG ERROR	MEM
MEM ARG ERROR-REQUEST EXCEEDS LIMIT	MEM
MEMBER FILE NOT IN MULTI-FILE SET (LABEL)	LABEL
Specified position number not found in multi-file set. Job terminated.	
MESSAGE LIMIT EXCEEDED	MSG
Number of messages issued by job exceeds installation maximum. Job terminated.	

MLRS TOO LARGE 1WS,1NW,1MT

Fatal error, code 22.

MMTC MEMORY PARITY ERROR 8T3

Fatal error, code 4.

MSG - ARG ERROR MSG

Address given to MSG is out of user's field length. Job terminated.

MT uu ALERT P4 1P4

1P4 detected a parity or alert status in the tape controller. (IP.DBUG=1 only)

MT uu BAC=xxxx 1R3

Current value of cell BAC is xxxx. Issued only when IP.DBUG=1.

MT uu BACKSPACE 1MT

Logical backspace was done. (IP.DBUG=1) (7-track only.)

MT uu BACKSPACE LP 1MT

Logical backspaced encountered load point. (IP.DBUG=1) (7-track only.)

MT uu BCxxxx 1R2

Byte count after backspace, forward read was at least xxxx. Issued only when IP.DBUG=1.

MT uu BLANK TAPE READ 1RS,4LB,4LC

Informative message. Channel was active and empty for 1 sec. Fatal error if detected by tape driver.

MT uu BUF SZ ER-TORET 1CT

Buffer does not have room for the record. PP hangs, waiting for non-zero control point error flag. A dump should be taken, if possible.

MT uu BYC=xxxx 1N2,1N3,
1R3,1NO

Last byte count read is xxxx. Issued only when IP.DBUG=1.

MT uu CONT BKSP 1P1,1P4

A controlled backspace was performed. (IP.DBUG=1 only)

MT uu CONTROLLED BACKSPACE 1MT

Controlled backspace done. (IP.DBUG=1) (7-track only.)

MT uu DCNS 1RV

Bit 0 of D.FNT+8 is not set on entry to 1RV. Issued only when IP.DBUG=1.

MT uu DCS 1RV

On entry to 1RV, device capacity exceeded (bit 0) is set in D.FNT+8 in the PP. Issued only when IP.DBUG=1. (7-track tapes.)

MT uu D.FTN+9=xxxx 1RT,1CT,1CS

Cell D.FNT+9 is xxxx. Issued only when IP.DBUG=1. (7-track)

MT uu DISCARD 1MT

A PRU was discarded as it did not contain expected information. (IP.DBUG=1) (7-track only.)

MT uu DISCARD P1 1P1

A PRU was discarded as it did not contain expected information (IP.DBUG=1)

MT uu DISCD NS NOISE 1N3

Non-standard noise record was discarded. Issued only when IP.DBUG=1.

MT uu DISCD ST NOISE 1N3

Standard noise record was discarded. Issued only when IP.DBUG=1.

MT uu EC=xxxx 1NO,1RV,1R2, 1R3, 1N3

Value of cell EC is xxxx. Issued only when IP.DBUG=1.

MT uu ERASE 1P2

A skip-bad-spot function was issued. (IP.DBUG=1 only)

MT uu ERROR MC=6 1RV

Bug exists in 1RV and error code is 6.

MT uu FET FMT ERR 4LB,4LC

Improperly formatted label in PP label buffer. Software error; notify site analyst.

MT uu IMPROPER TRAILER, EQU FORCED 1RP

EQUI trailer label after tape marks on SCOPE tape. System will simulate EQUI label.

MT uu LABEL PARITY ERROR 4LB,4LC

Parity error detected during attempted label read. Informative message.

MT uu LABELED ITO

Tape requested as unlabeled has a valid label. System waits for GO or RECHECK type-in.

MT uu LDC=xxxx 1NO,1N2,1N3

Value of cell LDCNT is xxxx. Issued only if IP.DBUG=1.

MT uu MT HDWR ERR 4LB,4LC

Failure to detect end-of-operation. Hardware failure notify site analyst.

MT uu NBRK xx 1MT

System noise record xx been written. (IP.DBUG=1) (7-track only.)

MT uu NO MSG BIT SET 1R2

Bit NM in byte flag of PP message buffer is set. Issued only when IP.DBUG=1.

MT uu NO ROOM FOR LABEL AT EOT 4LB,4LC

Failure in attempt to write trailer label because of parity errors. So much tape was erased that further operations may run off the end of the reel. 4LB aborts writing label.

MT uu NO WRITE ENABLE 1WI,1WS,4LB,4LC,1NW,1W9,1R2,1MT,1N9

A tape without a write ring cannot be written. Operator should clear and unload unit, insert write ring in reel, remount it, and ready the unit.

MT uu NOISE WARNING 1 1N2

NSLIM (set at 12) noise records were encountered in a read forward, bypassing standard noise records, trying to locate the next non-noise record. Last bad PRU number in T.TAPES is set to current PRU number. A new read forward loop is entered that will bypass all standard noise records without verification, (no reverse read to determine noise). Also the next multi-byte noise record is bypassed without verification if it occurs before a single-byte noise record or next record other a noise.

MT uu NOISE WARNING 2 1N2

The read loop following NOISE WARNING 1 failed to locate the next non-noise record. A new read loop will bypass all standard noise records and all multi-byte noise records without verification.

MT uu NOISE WARNING 3 1N3

Bypassing all multi-byte noise without verification failed to produce non-noise record. The next read loop will bypass all standard noise records and all multi-byte non-standard noise records without verification. If a single-byte noise record occurs before the next non-noise record, one single-byte noise record will be bypassed without verification.

MT uu NOISE WARNING 4 1N3

The read loop which occurred after NOISE WARNING 3 failed to produce a record other than noise. A read loop will be entered that bypasses all noise without verification.

MT uu NOT READY 1WI,1WS,1RT,1RS,4LB,1NW,1W9,1NR,1R9,1R2,1R3, 1N0,1N2,1N9,1N3,1P1,1P2,1P3,1P4,1MT,1TF,2TB

Job is trying to access a unit which is not ready. Operator should mount the proper tape and ready unit.

MT uu OP MODE P1 1P1

1P1 is trying to read the LGR in the opposite mode. (IP.DBUG=1)

MT uu PARITY P1 1P1

1P1 received a parity or alert status from the tape controller. (IP.DBUG=1)

MT uu PRU = xxxxxxxx

1RV,1R3,1P1,
1P2,1P3,1P4

Current PRU number is xxxxxxxx.

MT uu PRU ERR T.TAPES

1R2,1R3

Last bad PRU number in T.TAPES exceeds current PRU number.

MT uu RC=xxxx HE=yyyy

1R2

Cell RC=xxxx. HECT=yyyy. Issued only when IP.DBUG=1.

MT uu RD a bc dddd eeeeeeee

1MT

A read debug message as follows: (7-track only.)

a	F = forward; R = reverse
b	B = binary; C = coded
c	blank = no parity; P = parity error
d	byte count
e	first 2 bytes on channel after I/O

MT uu RD ERR DEV CAP EXC

1RV

Device capacity exceeded on the current record.

MT uu RD ERR HARDWARE LD

1R3

Solid lost data or transmission parity error occurred in the position forward routine in 1R3.

MT uu RD ERR LOST DATA

1RV

Lost data error on current PRU could not be recovered.

MT uu RD ERR NOISE IN IRG

1RV

Noise in preceding IRG record prevented read forward. Record was recovered with reverse read, but driver cannot guarantee that data can be turned around successfully in the buffer. Noise preceding record could cause a record to be lost.

MT uu RD ERR READ OPP MODE

1RV

The current record could not be read without error in the mode specified by the request. This record could be read without error in the opposite mode. The data is not guaranteed. (7-track tape.)

MT uu RD ERR REC FRAGMENT 1RV

Number of bytes read from SCOPE tape is not evenly divisible by 5 nor by 5 with a remainder of 4.

MT uu RD ERR TAPE PAR ERR 1RV

A tape parity error, CRC error, or multi-track error occurred and could not be recovered. (7-track tape.)

MT uu RD ERR XMSN PAR ERR 1RV

Transmission parity error could not be recovered. Call a C.E.

MT uu RD RVD DEV CAP EXC 1RV

Device capacity exceeded error was recovered. The record is good.

MT uu RD RVD LOST DATA 1RV

Lost data condition was recovered.

MT uu RD RVD NOISE IN IRG 1RV

Although preceding IRG has noise, it was read successfully.

MT uu RD RVD NOISE RECORD 1N2,1N3

Verified non-standard noise record was bypassed on tape.

MT uu RD RVD REC FRAGMENT 1RV

Record fragment error was recovered successfully; data is good.

MT uu RD RVD STAN NOISE 1N2,1N3

Non-standard noise record was verified as a standard noise record.

MT uu RD RVD TAPE PAR ERR 1RV

Tape parity error was recovered successfully; the record is good.

MT uu RD RVD XMSN PAR ERR 1RV

Transmission parity error was recovered successfully; record is good.

MT uu RDC=xxxx 1R3

Current value of cell RDC is xxxx. Issued only when IP.DBUG=1.

MT uu .RDF... 1NO

Read forward on tape. Issued only when IP.DBUG=1.

MT uu ...RDF 1R3

Record was read forward. Issued only when IP.DBUG=1.

MT uu RDF P1 1P1

Forward read was performed. (IP.DBUG=1)

MT uu RDF P3 1P3

Forward read was done. (IP.DBUG=1 only)

MT uu RDF P4 1P4

1P4 performed a forward read. (IP.DBUG=1 only)

MT uu .RDR... 1NO

READ reverse on tape. Issued only when IP.DBUG=1.

MT uu ...RDR 1R3

Record was read reverse. Issued only when IP.DBUG=1.

MT uu RDR P1 1P1

Reverse read was performed. (IP.DBUG=1 only)

MT uu RDR P3 1P3

Reverse read was done. (IP.DBUG=1 only)

MT uu RDR P4 1P4

1P4 performed a reverse read. (IP.DBUG=1 only)

MT uu REJECT 1WI,1NW,1WS,4LB,1W9,1RT,1RS,1NR,1R9,1R2,1R3,
1N0,1N2,1N3,1N9,1P1,1P2,1P3,1P4,1TF,2TB,1MT

Hardware is rejecting a function code. Call a customer engineer .

MT uu RESERVED   1WI,1WS,1RT,1RS,4LB,1W9,1NW,1NR,1N9,1R2,1R3,
1R9,1N0,1N2,1N9,1N3,2TB,1TF,1P1,1P2,1P3,1P4,1MT

The tape unit is reserved on another channel and should not be. Hardware or software malfunction occurred during tape input/output processing.

MT uu RET=xxxx   1RV

The value in cell RET is xxxx. Issued only when IP.DBUG=1.

MT uu RETRY=xxxx   1R3

Current value of cell RC is xxxx. Issued only when IP.DBUG=1.

MT uu S=xxxx FET=yyyy   1CT,1RT,1CS

Cell STATUS=xxxx. Cell D.BA+4=yyyy. Issued only when IP.DBUG=1.

MT uu SKP MFR nnnnnnnn   1N2,1N3

While searching for PRU nnnnnnnn in reverse direction, multi-byte noise record was bypassed withcut verification.

MT uu SKP SFR nnnnnnnn   1N3

A single-byte noise record was bypassed without verification while searching for PRU nnnnnnnn.

MT uu TSB=xxxx TIB=zzzz   1RV,1N2,1N3,
1NO,1R2,1R3

Value of cell TSB is xxxx. Value of cell TIB is zzzz. Issued only when IP.DBUG=1.

MT uu UNEXPIRED   4LB,4LC

User is trying to write on a tape with an unexpired label. System waits for GO or RECHECK.

MT uu UNLABELED   4LB,4LC

Tape requested as labeled does not have readable label. System waits for GO or RECHECK.

MT uu VER AND COMP   1P3

1P3 finished verifying that recovery in immediate area of recovered PRU.

MT uu WRT ERR BAD CBKSP 1P1,1P4

Invalid data was found (parity error or record of incorrect length) after a controlled backspace.

MT uu WRT ERR BAD CONTR BKSP 1MT

Invalid PRU (length or parity error) was found after a controlled backspace. (7-track only.)

MT uu WRT ERR BAD DSCRD 1P1

When reread, a record previously discarded as noise was greater than noise length.

MT uu WRT ERR BAD ERASE 1P2,1MT

Parity error was detected while erasing a tape.

MT uu WRT ERR CANT FND LGR 1P1

Last good data record was not found.

MT uu WRT ERR CANT FND SNR 1P4

A readable system noise record cannot be found.

MT uu WRT ERR DVR ERR 1P1,1P2, 1P3,1P4

An error detected in the write driver design; a dump should be taken. (IP.DBUG=1 only)

MT uu WRT ERR ERASE LIMIT 1P2,1MT

More than IP.SKCNT attempts to recover have been unsuccessful.

MT uu WRT ERR FL POS UNCRT 1P1

After a reverse read of a bad PRU, the tape appears to be positioned in the midst of the previous record.

MT uu WRT ERR FMK BFR LGR 1P1

During search for the last good record, encountered a mark which is not a filemark.

MT uu WRT ERR FMK BFR SNR 1P4

A filemark was found while searching backward for a system noise record.

MT uu WRT ERR HUNG FULL 1P2

Data channel hung full after an OAM. Informative message.

MT uu WRT ERR LGR BFR SNR 1P4

The last good record occurred before an expected system noise record.

MT uu WRT ERR LGR RRD BAD 1P1

The last good record reread was bad.

MT uu WRT ERR LOST DATA 1P1,1P2,1P3,1P4,1MT

Last data status returned by the hardware.

MT uu WRT ERR NO LGRL 1P1

The last good record length was not available for recovery. A substitute algorithm is used in this case. (IP.LGRL=1 only)

MT uu WRT ERR NOISE IN IRG 1P3,1MT

A PRU larger than noise record length was found which was not a valid data block.

MT uu WRT ERR POS ERR LPT 1P1,1P3,1P4

Load point has been encountered; position is uncertain.

MT uu WRT ERR POS UNCERTAIN 1MT

1MT lost its position; integrity of the PRU being recovered or previous PRU's cannot be guaranteed. (7-track only.)

MT uu WRT ERR POS UNCRT 1P1,1P3,1P4

Tape may be positioned in the midst of the last good record. May be preceded by a message which details the positioning error.

MT uu WRT ERR PSBL FRAGMNT 1P1

After a reverse read of a bad PRU, the tape appears to be positioned in the middle of the bad PRU.

MT uu WRT ERR REV EQ LGRL 1P1

A PRU read forward as noise length was same length as last good record when read in reverse.

MT uu WRT ERR SGL FRM NOZ 1P3

A single frame of noise was detected while verifying recovery.

MT uu WRT ERR XMSN PAR ERR 1P1,1P2,1P3,1P4,1WI

Hardware error.

MT uu WRT ERR 25 FT ERASED 1P2

50 unsuccessful attempts were made to recover the error.

MT uu WRT NZ 1P2

A system noise record was written. (IP.DBUG=1 only)

MT uu WRT RVD 1P3

A write error was recovered.

MT uu W1=xxxx W2=yyyy 1NO,1N3,1R3

First byte in PP buffer is xxxx; second byte is yyyy. Issued only when IP.DBUG=1.

MT uu XMSN P.E. 4LB

MT uu XMSN P.E. 4LB,4LC

Transmission parity error occurred during tape I/O. Hardware failure; notify site analyst.

MT uu XMSN PARITY ERROR 1WS,1RT,1RS,1W9,1W1,1NW,1R9,1NR,1R2,1R3,1NO,1N2,1N3,1P1,1P2,1P3,1PA,1MT,2TB,1TF

Hardware malfunction.

MT uu yy Sz 1P3

Error of type yy was detected by section z of 1P3. (IP.DBUG=1 only)

yy =	XM	transmission parity error
	PR	tape parity error
	MP	MMTC memory error
	LD	lost data error
	NT NZL	not noise length error
	P2 PAR	1P2 in rerouting the PRU deleted parity error
	P2 LSD	1P2 in rerouting the PRU deleted last data
	P2 CHF	1P2 in rerouting the PRU deleted channel hung full
	P2 XMP	1P2 in rerouting the PRU deleted transmission parity
	BAD LGTH	a length error error exists
z =	0-5	

MT uu 1CS LOADING IRS 1CS

1CS is loading 1RS. Issued only when IP.DBUG=1.

MT uu 1CS LOAD 1RP 1CS

1CS is loading 1RP. Issued only when IP.DBUG=1.

MT uu 1CS LOAD 1TF 1CS

1CS is loading CIO to load 1TF. Issued cnly when IP.DBUG=1.

MT uu 1CS RET TO USER 1CS

1CS has set the complete bit and is dropping. Issued only when IP.DBUG=1.

MT uu 1CT LOAD 1RP 1CT

1CT is loading 1RP. Issued only when IP.DBUG=1.

MT uu 1CT LOAD 1TF 1CT

1CT is loading CIO to load 1TF. Issued cnly when IP.DBUG=1.

MT uu 1CT LOADING 1RT 1CT

1CT is re-loading 1RT. Issued only when IP.DBUG=1.

MT uu 1CT RET TO USER 1CT

1CT is setting complete bit and dropping PP. Issued only when IP.DBUG=1.

MT uu 1NO... 1NO

1NO is loaded to examine noise on tape. Issued only when IP.DBUG=1.

MT uu 1N2... 1N2

1N2 has been loaded. Issued only when IP.DBUG=1.

MT uu 1N3... 1N3

1N3 has been loaded. Issued only when IP.DEUG=1.

MT uu 1RT LOAD 1RP 1RT

1RT is loading 1RP. Issued only when IP.DBUG=1.

MT uu 1RT LOAD 1TF 1RT

1RT is loading CIO to load 1TF. Issued cnly when IP.DBUG=1.

MT uu 1RV F BYC=xxxx 1RV

1RV is loaded, at least the last read was forward, and byte count read was xxxx. Issued only when IP.DBUG=1.

MT uu 1RV R BYC=xxxx 1RV

1RV is loaded, at least the last read was reverse, and byte count read was xxxx. Issued only when IP.DBUG=1.

MT uu 1R2... 1R2

1R2 is loaded. Issued only when IP.DBUG=1.

MT uu 1R3... 1R3

1R3 is loaded. Issued only when IP.DBUG=1.

N EQUAL INVALID CHAR COPYCR,COPYCF, COPYBR,COPYBF

Number of records/files to be copied is zero, a letter, or special character on the copy control card.

NAME IN SPECIFICATION FIELD TRUNCATED TO 7 CHARACTERS SEGBILD

NAME LIST MAY NOT CONTAIN AN INTERVAL EDITLIB

NAME LIST TOO LARGE - 40 NAME MAXIMUM EDITLIB

NAME OF RECORD IS MISSING OR DELIMITER FOLLOWS A DELIMITER EDITLIB

NAME TOO LARGE - DELIMITER IS MISSING EDITLIB

NEEDED 2 LEVEL NUMBERS BELOW 64 LOADO

NEW ECS FE TOO SMALL. REQUEST xxxx (RESTART) RESTART

xxxx is required ECS field length for the jcb.

NEW FL TOO SMALL RFL, xxxxxx. (RESTART) RESTART

Field length of Restart must not be less than field length at Checkpoint. xxxxxx is required field length for the job. Job terminated if RESTART memcry request cannot be satisfied.

NEW PACK NOT SAME AS OTHERS IN FAMILY 1DA

PACK type-in, in response to RPACK (pname,N) request, generated by system when a pack family is full, specifies a disk pack whose EST type code (AP or AM) is not the same as that of other packs in family.

NEWCYCLE CATALOG PFC

NO AVAILABLE DEVICE OF TYPE REQUESTED REQ

User requested specific device type not in this system; job terminates.

NO CHECKPOINT TAKEN 1RC

CKP aborted.

NO CHECKPOINT TAKEN CKP

CKP terminated because of CALL error.

NO CONTINUATION CARD REQ

No terminator found on REQUEST card; REQUEST aborts.

NO DAYFILE FOUND 1DF

1DF could not find FNT entry for DAYFILE - system malfunction.

NO E OR N ON CTRL CARD 1DA

Sent to both dayfiles if RPACK control card does not contain E or N as second or third parameter; job terminated.

NO END OF OP 6WM

When end-of-operation is not returned, 6WM is loaded by all tape drivers except 1RV,1CT,1CS,1CR,1C9 to issue this message.

NO -INPUT- FILE LOC

NO INTERCOM ID FOR CENTRAL SITE JOB DSP

Central job must specify INTERCOM terminal to which the file should be disposed.

NO LFN OR PFN PFM

Neither a logical file name nor permanent file name were specified.

NO OUTPUT FILE ON THE COPYN CONTROL CARD COPYN

COPYN control card must specify an output file.

NO OVERLAY DIRECTIVE BEFORE FIRST PROGRAM LOADO

NO PERMISSION PFA

Attach request established no permissions for the file request.

NO PFD DEVICE PFA,PFC, PFE,PFP

Installation error. Report message to a system analyst.

NO PROGRAMS SPECIFIED ON SLOAD LOAD

NO RBR FOR EST 00 1EJ

NO RBR FOR THIS PACK 5DA,3PK,1DA

RBR table for one of the packs could not be found; 5DA indicates a system error.

RBR table for a new file on sequential pack could not be found; 3PK indicates a system error.

BLANK, DEVADD, UNLOAD, or PACK type-in specified disk pack in the EST, but no RBR can be found for it. 1DA indicates a system error; job terminated.

NO RERUN BIT SET, JOB CANNOT BE RERUN 6SI

Parity error occurred when job was being rolled in. Job was terminated instead of rerun because the no-rerun bit was set.

NO ROOM FOR EXTRA CYCLE	PFC
Permanent file has 5 cycles.	
NO SUCH FILE NAME	5DA
File named by REMOVE control card does not exist at the control point.	
NO SUCH PRIVATE PACK	5DA
NO SUCH PROGRAM CALL NAME - (name)	LOAD
Name on a program call control card does not match any legal possibility: Local file name PP program name Entry point of a control card callable program in one of the libraries defined by the global library set.	
NO TERMINATOR ON CONTROL CARD	LOAD
NO TERMINATOR ON DIRECTIVE	LOAD
NO TRANSFER ADDRESS	LOADERE, LOAD
At completion of load, no program provides transfer address.	
NO VALID STARTING ENTRY	SEGBILD
NO VSN CARDS FOUND	1MH
Tape job has no VSN control cards in control card record.	
NO XFER TABLE	LOADO
NO //END// STATEMENT	SEGBILD
NON-EXISTENT LIBRARY GIVEN - (libname)	LOAD
NON-FATAL ERROR(S) IN OVERLAY GEN.	LOADO
NOT ALLOWED TO DROP EDITLIB	MDI
Once GO has been typed, EDITLIB must complete its operation. Job continues.	

NOT CONTROL-CARD-CALLABLE - (name) LOAD

A program call control card matches the name of an entry point of a library program defined by the global library set. The program for use, however, cannot be called by control card; it is intended as a subroutine or a higher level absolute overlay.

NOT DISK PACK 1DA

BLANK, DEVADD, UNLOAD or PACK type-in gives EST ordinal for other than pack; job terminated.

NOT ENOUGH FIELD LENGTH FOR TRAP TABLES TRAPPER

NOT ENOUGH FL, NO DEBUG DONE TRAP

NOT ENOUGH FL TO LOAD RESTART RESTART

RESTART FL must be 25000 or greater, depending on FL of checkpoint job.

NOT IN SYSTEM JDP

The prefix of a tape message indicates the kind of tape affected. MT uu indicates 7-track tape, NT uu indicates 9-track tape. Many messages are the same for both kinds of tape except for the prefix. In this appendix, all messages common to both types of tape, as well as those that apply to 7-track, are printed with the MT prefix. Messages printed with the NT prefix apply only to 9-track tape.

NT uu BAC=xxxx 1R3

Current value of cell BAC is xxxx. Issued only when IP.DBUG=1.

NT uu D.FNT+9 IS xxxx 1CR

Cell D.FNT+9=xxxx. Issued only when IP.DBUG=1.

NT uu RD ERR DEV CAP EXC 1RV

Device capacity has been exceeded on the current record.

NT uu RD ERR LOST DATA 1RV

Lost data error on current PRU could not be recovered.

NT uu RD ERR MMTC MEM PAR 1RV

MMTC memory parity error during current PRU read could not be recovered.

NT uu RD ERR NOISE IN IRG 1RV

Noise in preceding IRG prevented read forward. Record could not be recovered with a reverse read.

NT uu RD ERR REC FRAGMENT 1RV

Number of bytes read from SCOPE tape is not evenly divisible by 5 nor by 5 with a remainder of 4.

NT uu RD ERR TAPE PAR ERR 1RV

Tape parity error, CRC error, or multi-track error on current PRU could not be recovered.

NT uu RD RVD NOISE RECORD 1N2,1N3

Verified a non-standard noise record as ncn-standard; record is discarded.

NT uu RD RVD STAN NOISE 1N2,1N3

Verified a non-standard noise record as standard noise; record is discarded.

NT uu RDC=xxxx 1R3

Current value of cell RDC is xxxx. Issued only when IP.DBUG=1.

NT uu ...RDF 1R3

Record was read forward. Issued only when IP.DBUG=1.

NT uu RET=xxxx 1RV

Value in cell RET is xxxx. Issued only when IP.DBUG=1.

NT uu RETRY=xxxx 1R3

Current value of cell RC is xxxx. Issued only when IP.DBUG=1.

NT uu S=xxxx FET=yyyy 1CR

Cell STATUS=xxxx. Cell D.BA+4=yyyy. Issued only when IP.DBUG=1.

NT uu SKP MFR xxxxxxxx 1N2,1N3

Skipped a multi-byte record without verification.

NT uu 1CR LOAD 1TF 1CR

1CR is loading CIO to load 1TF. Issued cnly when IP.DBUG=1.

NT uu 1CR LOADING 1NR 1CR

1CR is reloading 1NR. Issued only when IP.DBUG=1.

NT uu 1CR LOADING 1RP 1CR

1CR is loading 1RP. Issued only when IP.DBUG=1.

NT uu 1CR RET TO USER 1CR

Overlay 1CR set complete bit in the FET and dropped PP. Issued only when IP.DBUG=1.

NT uu 1N2... 1N2

1N2 has been loaded. Issued only when IP.DBUG=1.

NT uu 1R3... 1R3

1R3 is loaded. Issued only when IPDBUG=1.

NT uu 6WM ER 1CR RC 1CR

6WM returned control to 1CR when 1CR tried to have 6WM issue a fatal error message. (ERRN8)

NEW IDENT ON CHANGE CARD IS ALREADY KNCWN UPDATE

An attempt was made to change an ident tc one already in existence. Fatal error.

NO ACTIVE CARDS WERE FOUND WITHIN THE COPY RANGE. NULL COPY UPDATE

Nonfatal error, UPDATE continued.

NO DECK NAME ON DECK CARD UPDATE

Fatal error message.

NO INPUT FILE, Q MODE, UPDATE ABORTED UPDATE

NO OLDPL, NOT CREATION RUN, UPDATE ABORT UPDATE

NUMBER OF RECORDS GREATER THAN 777777B Second parameter of a skip control card exceeded maximum number of allowable records.	SKIPF,SKIPB
NUMBER EXCEEDS 5-DIGIT LIMIT 99999 USED	EDITLIB
OBJECT DECK CONTAINS ONLY A PREFIX TABLE	EDITLIB
OCC ERROR - Correction card format is incorrect.	LOC
OCTAL FIELD CONTAINS DECIMAL DIGITS	EDITLIB
OCTAL FIELD CONTAINS A NAME	EDITLIB
ODD NUMBER OF NAMES IN ((SUBST)), LAST ONE DROPPED	LOADO,SEGBILD
ODD WORD COUNT IN ENTR/REPL TABLE	SEGBILD
OLD OR NEW PARAMETER MISSING / DELIMITER FOLLOWS A DELIMITER.	EDITLIB
OLDLIB DIRECTIVE MUST IMMEDIATELY FOLLOW LIBRARY DIRECTIVE	EDITLIB
OLDLIB DIRECTIVE IS VALID WHEN BUILDING A NEW USER LIBRARY	EDITLIB
OLDPL READ ERROR - POSSIBLE LOST DATA AFTER FOLLOWING	UPDATE
ONLY LFN SYSTEM CAN BE MODIFIED	EDITLIB
ONLY EDITLIB MAY CALL MDI Because it can modify the running system, MDI must be protected from unauthorized calls. In this case a program not loaded totally from the system library, (not EDITLIB) tried to issue an MDI request.	MDI
OPERATOR DROP	1EJ
OPTION INVALID WITH RANDOM OLDPL OR SEQUENTIAL NEWPL	UPDATE
OPTION INVALID WITH SEQUENTIAL OLDPL	UPDATE

OVERLAY DIRECTIVE NOT FIRST	LOAD
During overlay generation, nothing may precede an OVERLAY directive.	
PACKS uu,uu,uu,....RELEASED - TYPE n.DROP	1EJ
PACK FILE COUNT ALREADY 0	5DA
In trying to fulfill a REMOVE card, system found the file and pack family, but file count is already 0. Indicates a system error.	
PACK NOT PUBLIC AND EMPTY	1DA
PACK LABEL NOT BLANK	1DA
DEVADD or PACK type-in responding to RPACK (pname,N) control card or request from system, specified disk pack in unloaded status with a private label.	
PARAM LIST TOO LONG (CKP)	CKP
The value of n in the user parameter exceeds 42 (decimal). Job terminated.	
PARAMETER COUNT IS GREATER THAN 4	SKIPF,SKIPB
Too many parameters on skip control card.	
PARAMETER WORD OUTSIDE USER FL	REQ
REQ parameter list is outside user's FL; job terminates.	
PARAMETER ERROR IN STS	STS
Invalid parameter supplied in STS call.	
PARAMETER F OR P INVALID WHEN RECORD NAME PRESENT	EDITLIB
PARAM LIST TOO LONG (CKP)	CKP
PARAM(S) MISSING ON ABOVE TRAP DIRECTIVE	TRAP
PARAMETER EXCEEDS 9 CHAR	PFCCP

PARITY ERROR CEM,6WM

Parity error occurred during read of an ECS buffered file, error may be in ECS or on RMS device.

PARITY ERROR IN ROLLIN 6SI

Job is automatically rerun if no-rerun bit is clear; otherwise the job terminates.

PARTIAL EDITLIB PERFORMED - SYSTEM MAY BE INOPERATIVE EDITLIB

PF ABORT PFM

PPU abort.

PF ALREADY ATTACHED PFA

Permanent file specified by permanent file name, ID, and cycle number has already been attached by this job.

PF CONTROL CARD HAS NO TERMINATOR PFCCP

PF control card has no terminator and there are no control cards following it.

PF CYCLE NO. = nnn PFA

If LC was specified, no CY was specified, or the CY value was not usable, the generated cycle number nnn is printed.

PF ERROR-CATALOG 1DF

Attempt to catalog new cycle of DAYFILE failed.

PF ERROR-PURGE 1DF

Attempt to purge old cycle of DAYFILE failed.

PF ERROR-ATTACH 1DF

Attempt to attach latest cycle of DAYFILE failed.

PF REQUESTED. SHOULD NON-PF xx BE ASSIGNED REQ

User REQUEST is for permanent file device; operator assigned non-PF device.

PFD DEVICE DOWN PF Routines

Message appears on system dayfile only.

PFD FULL — PFC,LPF

All mass storage space allocated to permanent file directory (PFD) has been used. Installation action required.

PFM STOPPED BY SYSTEM — PFM

PFM SYSTEM ERROR — 1PC

Permanent file manager system error, issued with one of the following messages:

BAD PF FNT POINTER
BAD PFD POINTER
FILE RELEASE COUNT BAD
BAD APF ORDINAL
BAD APF COUNT
CYCLES DISAGREE
RBT-RBTC DISAGREE
CYCLE NOT FOUND
BAD FNT FET ADDRESS

PFx SYSTEM ERROR CODE xx — PF Routines

System error occurred in permanent file rcutine PFx. Reliability of permanent file manager is in doubt. Error type is indicated by code xx:

1	PFD entry count error
2	RBTC entry length in error
3	Cycle not found
4	First word of RBTC PRU in error
5	Bad PFD FNT
6	Bad RBTC FNT
7	Bad status returned in PFT FNT
8	Purge-on-not-in-use entry
9	Bad PFA cycle search
10	System error, bad FNT
11	Bad PFD pointer in RBTC
12	Bad cycle number in RBTC
13	Interlock problem
14	Bad RBTC pointer
15	Duplicate APF entry
16	Bad LPF communication
17	Illegal function attempted on incomplete cycle
18	CM chain shorter than RBTC chain
19	RBT-RBTC chains are not equal
20	Bad PRU index

PFN EXCEEDS 40 CHARS   PFCCP

PFN IS (permanent file name)   PFM

If the permanent file name was assumed, the name will be printed as part of the message.

PHYSICAL/LOGICAL POSITIONS DISAGREE   1TO,6WM

Information only, no error code. Tape file is not physically positioned at load point, although logical PRU count indicates that it should be.

1R2 ENCOUNTERED LOAD POINT AFTER BACKSPACE   1R2

PIDL TABLE SEEMS TO BE MISSING   SEGBILD

PN=pname, VN=visum, JD=juldate   1DA
PAUSING FCR GO OR DROP

Sent to both dayfiles and the B display if pack referenced by n.BLANK,uu. type-in has a private label. pname is logical pack family name, visum visual is serial number, and juldate Julian date on the label.

PNT OVERFLOW   EDITLIB

More than 2047 programs in program name table.

PNUT OVERFLOW   EDITLIB

Indicates EDITLIB program error.

PP CALL ERROR   1EJ

Job terminated because central processor requested PP program with name that did not begin with a letter.

PP PROGRAMS NOT IN PPLIB CANNOT BE ACCESSED BY PP RESIDENT   EDITLIB

PP PROGRAM ALREADY IN PP LIBRARY   EDITLIB

PP PGM NAME TABLE IS EMPTY   EDITLIB

PP 0   MTR

System fatal error.

PPLIB IS BOTH SOURCE AND DESTINATION LIBRARY   EDITLIB

PREFIX TABLE MISSING OR RECORD IS NOT AN OBJECT DECK   EDITLIB

(PREVIOUS ERROR CONDITION RESET)   RPV

RPV has reset the error flag and user exchange package.

PRIVATE PACK(S) RELEASED   1EJ

PROCESSING DIRECTIVE NUMBER xx   EDITLIB

PROG. CONTAINS ILL-STRUCTURED TABLE	LOADO
PROG. ENDS AFTER 77-TABLE	LOADO
PROG. DOESNT BEGIN WITH PIDL TABLE	LOADO
PROGRAM LGTH. IN PIDL TABLE IS NEGATIVE	SEGBILD
PROGRAM NAME HAS ALREADY OCCURRED	SEGBILD
PROGRAM MAX REACHED. ADJUST MAXLPPNT IN EDITLIB - INSTALL NEW VER.	EDITLIB
PUBLIC PACK NOT EMPTY UNLOAD type-in specified public pack in which information is stored.	1DA
P2 IS NOT IN THE FILE OR IS UNDEFINED	COPYN
P2 NUMERIC EXTENDS BEYOND DOUBLE EOF Numeric p2 on record identification card indicates a logical record beyond double end-of-file that terminates file.	COPYN
RBTC FULL All mass storage space allocated to permanent file table, RBTC, has been used. Installation action required.	1FC
READ OR SKIPF AFTER WRITE Return dependent on EP bit setting. Error code 30. Attempt was made to read or skip data on a file positioned after a newly written record.	CIO,1MF
READ PERMISSION NOT GRANTED Fatal error, code 22. Read function issued to a file for which read permission was not granted.	CIO
READING FILE (filename)	SEGBILD,LOADO
READY MUST PRECEDE A COMPLETE	EDITLIB
READY MUST PRECEDE A INCLUDE	EDITLIB
READY MUST PRECEDE A INCLUDEP	EDITLIB
READY MUST PRECEDE A CHANGE	EDITLIB
READY MUST PRECEDE A REMOVE	EDITLIB

READY OR LIBRARY MUST PRECEDE A MOVE	EDITLIB
READY MUST PRECEDE A TRANSFER	EDITLIB
READY MUST PRECEDE A LISTLNT	EDITLIB
RECOVERED RMS ERROR FILE NAME xxxxxxx FET ADDRESS xxxxxx STATUS ssss cccc EQ ee ADDR aaaa bbbb TRY nn RBRxx RByyyy PRUzzzz RECORDED ADDR pppp qqqq	1SX
Mass storage device error was corrected after nn (octal) attempts to read/write a PRU. Controller status ssss, converter status cccc, equipment number ee, physical address of the PRU aaaa bbbb, logical address xx, yyyy, zzzz and recorded physical address pppp qqqq are all octal. Last line appears only for the 6603-II and 6638 disk units. Only the first three lines appear in the job dayfile. Non-fatal - informative.	
RELEASE ILLEGAL ON PERMANENT FILE	4ES,CIO
Fatal error code 22. Code return is dependent on error processing bit setting.	
RELOCATABLE PGM NAME x...x DOES NOT HAVE AN ENTRY POINT	EDITLIB
REMAINDER OF CARD SKIPPED	EDITLIB
REPL TRIED TO FETCH FROM OUTSIDE OVERLAY AS BUILT SO FAR	LOADO
REPRIEVE ABORTED CHECKSUM BAD	RPV
An attempt was made to reprieve job after an error, but recovery routine's checksum was in error.	
REPRIEVE ROUTINE NOT IN FL	RPV
REPRIEVE ABORTED NOT IN FL	RPV
REPRIEVE ABORTED SYSTEM ERROR	RPV
REQUEST ABORTED	REQ
REQUEST FOR ARCHIVED FILE - WAITING FOR CENTRAL OPERATOR GO OR DROP	1IM
REQUIRES LARGER BLANK THAN DEFINED ON LOWER LEVEL - REQMNT. IGNORED	
e.g., the 0,0 overlay declares blank common as 10000 locations, and the 1,0 overlay declares it as 20000.	

REQUIRED OCTAL PARAMETER IS MISSING	EDITLIB
RERUN FORCED BY DEADSTART RECOVERY	TDS
Job caught at a control point during deadstart recovery.	
RESET NOT DEFINED FOR A USERS EDITLIB	EDITLIB
RESIDENCY PARAMETER -ECS- INVALID DURING DEADSTART CREATION	EDITLIB
RESTORE NOT DEFINED FOR A USERS EDITLIB	EDITLIB
RESTART NUMBER TOO BIG (RESTART)	RESTART
Requested checkpoint number greater than last checkpoint on tape.	
RESTART POSITION UNCERTAIN - RUN ABORTED	RESTART
EOI/EOR/EOR encountered before tape is repositioned correctly.	
RET=xx STATUS=yyyy	1CR
Value of cell RET is xx. Value of cell STATUS is yyyy. Issued only when IP.DBUG=1.	
RETENTION PERIOD TOO LONG, MAX USED	PFR,1FC
RETURN, REWIND, OR UNLOAD MUST HAVE AT LEAST ONE PARAMETER	RETURN
REWRITE REQUIRES MODIFY PERMISSION	4ES
Fatal error code 22.	
RFL,1200.(CKP)	CKP
Checkpoint needs at least 1200 word FI.	
RP = nnn DAYS	1FC,PFR
Default or maximum retention period of nnn days was used.	
RPV CALL WITHOUT AUTO-RECALL	RPV
RPV UNABLE TO RESTORE PREVIOUS ERROR	RPV
User request to restore error flag and exchange package not carried out. User request is usually done by subroutine RECOVR$.	
S=xxxx FET=yyyy	1NR,1RS
Cell STATUS contains xxxx. Cell D.BA+4 contains yyyy. Issued only when IP.DBUG=1.	

SECONDARY DOESNT FOLLOW ITS OWN PRIMARY LOADO

SEGLOAD CARD ERROR LOAD

Keyword identifies card as specifying a SEGLOAD operation, but the card has a format error.

SIS OR SDA PERMISSIONS VIOLATED 4ES

If extended error processing and an FET extension are specified, the above message appears in the job and system dayfiles for a SIS or SDA file if the permission bits in the FST do not match those in the extended FET.

SITE INDICATOR AFTER = MUST BE C, I, OR E DISPOSE

Site indicator (k) on DISPOSE card musta be C (central site), I (INTERCOM terminal) or E (EXPORT/IMPORT terminal). DISPOSE card will be ignored.

SLOAD PROGRAM NOT FOUND - (name) LOAD

Program specified was not found on the file.

SPECIFIED DEVICE NON-EXISTENT CIO,3DO,REQ

Fatal error code 22.

SPECIAL CKP

CKP called by console type-in.

STATUS AREA OUTSIDE FL STS

Address of the status area as specified in the STS call is outside the job's FL.

SUBST FORMAT ERROR LOAD

SUCH NAMES FORBIDDEN TO PR.PACK FILES 5DA

A REQUEST for a new file on a pack family, is OUTPUT, PUNCH, etc., which causes the file to have a disposition code at job termination. Disposition codes are not allowed for family pack files.

SWAPIN PARITY ERROR, FNTS LOST - TYPE GO 6SI

Parity error during swap-in caused loss if job FNT's. If job had attached permanent files, DUMPF and TRANSPF utilities may not run until after a deadstart recovery.

SWAPIN PARITY ERROR, FNTS RESTORED 6SI

SWAPIN PARITY ERROR, NO RERUN BIT SET 6SI

Swapin parity error occurred; operator could not rerun job because the no-rerun bit was set.

SWAPIN PARITY ERROR, REREAD JOB - TYPE GO 6SI

Swapin parity error occurred and the no-rerun bit was not set. Operator should reread job into computer if possible.

SYSTEM ERROR, CANT FIND LOADER 1AJ

Entry point LOAD not in CM resident library.

SYSTEM ERROR TAPES-TABLE 1MF,1CL,1RP,REQ,2TB,1NR,1TO,1RS,1RT,1R9,
1WI,1WS,1WN,1W9,1MT

Fatal error, code 22. EST ordinal of tape unit assigned to file could not be located in TAPES table in CMR. Call a CDC analyst.

SYSTEM ECS PARITY ERROR, FATAL CEM

Parity error in ECS descriptor information occurred during file usage. Job terminated.

SYSTEM ECS PARITY ERROR CEM

Parity error in ECS descriptor information occurred during file release. Not fatal to job.

SYSTEM EDITLIB NOT PERMITTED THROUGH INTERCOM MDI

EDITLIB and INTERCOM are not compatible; terminal user attempting system EDITLIB is aborted.

SYSTEM COMMUNICATION ERROR (1SP...1SX) 1SX

1SX called with invalid parameters. Fatal error.

TABLE OVERFLOW ERROR - EDITLIB REQUIRES MORE FI	EDITLIB
TABLE OVERFLOW WHILE PERFORMING A DELETE. ENLARGE BUF3	EDITLIB
TABLE WOULD STORE IN ABSOLUTE OR NEG. RELOC.	SEGBILD
TAPE FORMAT ERROR, NOTIFY SYSTEMS Queue name in tape label does not correspond to type-in entry. Drop job and start again with correct tape or type-in.	XXXRESQ
TAPE xx OFF OR ASSIGNED Response to a UL xx type-in, where xx is tape unit EST ordinal.	1MH
TAPE xx NOT READY Response to a UL xx type-in.	1MH

TAPE LIMIT EXCEEDED (CKP) xxxxxxx — CKP

Number of tapes limited to 16.

TEXT CARD CONTAINS AN ILLEGAL SEPARATOR — COPYN

Only acceptable separators are + - , ( ) or blanks.

TEXT OR REPL TABLE GOES BEYOND PROG. BLOCK LGTH. — SEGBILD

THE ABOVE CARD AFFECTS A DECK OTHER THAN THE DECLARED DECK — UPDATE

Non-fatal error. The card in question will be ignored.

THE ABOVE CARD IS ILLEGAL DURING A CREATION RUN — UPDATE

Fatal error.

THE ABOVE CONTROL CARD IS ILLEGAL AFTER A DECK HAS BEEN DECLARED — UPDATE

Non-fatal error.

THE ABOVE OPERATION IS NOT LEGAL WHEN REFERENCING THE YANK DECK — UPDATE

Fatal error.

THE ABOVE SPECIFIED CARD WAS NOT ENCOUNTERED — UPDATE

- OR -

THE TERMINAL CARD SPECIFIED ABOVE WAS NOT ENCOUNTERED IT MAY BE IN A DECK NOT MENTIONED ON A COMPILE CARD — UPDATE

This message is printed as part of unprocessed modifications. The third line is printed when unprocessed modifications are found in a Q mode UPDATE. Fatal error message.

THE INITIAL CARD OF THE COPY RANGE WAS NOT FOUND. NULL COPY — UPDATE

Non-fatal error.

THE TERMINAL CARD OF THE COPY RANGE WAS NOT FOUND. COPY ENDS AT END OF SPECIFIED DECK — UPDATE

Non-fatal error.

THE FOLLOWING BLOCKS HAVE BEEN PREVIOUSLY DEFINED SMALLER OR IN DIFFERENT STORAGE. OLD DEFNS. HOLD.	LOADO
THE MESSAGE LEVEL PARAMETER NOT AVAILABLE FOR A USER EDITLIB	EDITLIB
THE LFN SPECIFIED IS NOT A DEADSTART TAPE	EDITLIB
THE ABOVE LISTED CARDS CANNOT EXIST IN THE YANK DECK AND HAVE BEEN PURGED DURING EDIT	UPDATE
THE FNT IS FULL	5DA
REQUEST for a new file on pack family cannot be satisfied.	
THE FNT IS FULL	1DA
No room in FNT for a new entry or existing member of pack family assigned in response to RPACK control card.	
THIS 2-LEVEL OVERLAY FOLLOWS A 0-LEVEL OVERLAY	LOADO
THIS DIRECTIVE NOT ON ONE CARD OR THERE IS NO TERMINATING DELIMITER	EDITLIB
THIS PACK STATUS NOT UNLOADED	1DA
BLANK, DEVADD, or PACK type-in specified disk pack without unloaded status; job terminated.	
TIME LIMIT	1EJ
Job exceeded CP time limit requested on job card. Job terminated.	
TOO MANY INPUT FILE NAMES ON COPYN	COPYN
Maximum is 10.	
TOO MANY TEXT CARDS IN THE INPUT RECORD	COPYN
Maximum is 13 or more, depending on the total number of characters on the card.	
TOO MANY CHBS -- INCREASE L.CHB	UPDATE
Correction history bytes exceeded the specified limit of 100 octal for a card. Job terminated.	
TOO MANY PARAMS IN EXECUTE REQUEST	LOAD
Maximum is 42.	

TOO MANY PARAMETERS (LISTMF)	LISTMF
More than allowed number of parameters on control card. Job terminated.	
TOO MANY FILES WITHIN JOB - 4 MAX	EDITLIB
TOO MANY PARAMETERS ARE GIVEN OR INVALID DELIMITER WAS USED	EDITLIB
TOO MANY WHENS ON TRACK DIRECTIVE	TRAP
TRANSFER DIRECTIVE VALID ONLY IN DEADSTART CREATION MODE	EDITLIB
TRAP CARD PARAM ERROR, NO DEBUG DONE	TRAP
TRAP DIRECTIVE ERROR(S)	TRAP
TRIED TO WRITE A NOISE RECORD	1MT,1WS, 1NW
Fatal. No error code.	
TRIED TO PUT SOMETHING BEYOND GIVEN LWA OF PROGRAM BLOCK	LOADO
TRIED TO PUT SOMETHING BELOW FWA OF OVERLAY	LOADO
TRIED TO PUT SOMETHING IN BLANK COMMON, ABSOLUTE, OR NEG. RELOC.	LOADO
TRIED TO PUT SOMETHING BETWEEN CM AND ECS	SEGBILD,LOADO
TRIED TO LOAD INTO BLANK COMMON	SEGBILD
TRIED TO LOAD INTO GLOBAL BL.x...x IN ANOTHER SEGMENT	SEGBILD
TRIED TO LOAD INTO ABS. OR NEG. RELOC.	SEGBILD
TRIED TO LOAD BEYOND LWA+1 OF SEGMENT	SEGBILD
TSB=xxxx TIB=yyyy	1CR
These values are issued only when IP.DBUG=1.	
TYPE GO, DROP OR RECHECK	JDP
UBC TOO LARGE	CIO
Unused bit count is greater than 59 in the S or L tape control word or in the seventh word of the FET.	
UNABLE TO LOCATE PGM x...x ON LOCAL FILE NAME	EDITLIB

UNCORRECTABLE PARITY ERROR 1SX

```
    FILE NAME xxxxxxx
    FET ADDRESS xxxxxx
    STATUS ssss cccc
    EQ ee ADDR aaaa bbbb TRY 76
    RBRxx RByyyy PRUzzzz
    RECORDED ADDR pppp qqqq
```

Return dependent on EP bit setting. Errcr code 04 returned in bits 9-13 of FET code/status. An uncorrectable device error was not recovered after 76 (octal) attempts to read/write a PRU. This error may be an actual parity error, lost data, disk mispositioning, dropping ready during data transmission address errors and defective track error. Controller status ssss, converter status cccc (for 6603, 6638, and 865), equipment number ee, physical address of bad PRU aaaa bbbb, logical address xx, yyyy, zzzz and recorded physical pppp qqqq are all octal. Last line appears only for 6603-II and 6638 disks units; aaaa bbbb not equal to pppp qqqq, indicates a positicning error. Only the first three lines appear in the job dayfile.

UNIT HUNG BUSY 6WM

When a unit hangs in a busy status, all I/O drivers (except 1RV, 1CT, 1CS, 1C9, 1CR) load 6WM to issue this message.

UNKNOWN IDENT NAME xxxxxxxxx UPDATE

A correction control card referenced an IDENT not in directory. Fatal error.

UNKNOWN LFN PFM

Logical file name specified or assumed nct found as a local file to this job.

UNLABELED OLDPL UPDATE

UNLOAD NOT ALLOWED ON INPUT. ICL

UNLOAD type-in specified a disk pack status that is not public.

UNRECOGNIZABLE DISP CODE DSP

Disposition code x for DISPOSE function must be a known disposition mnemonic. DISPOSE function will be ignored.

UNRECOGNIZABLE ID (last param)	DSP
Identifier y for DISPOSE function must be 1 or 2 alphanumeric characters. DISPOSE function will be ignored.	
UNRECOGNIZABLE BINARY TABLE TYPE	LOADO
UNRECOGNIZABLE TABLE FOLLOWING A PREFIX TABLE	EDITLIB
Non-loader table encountered in a binary record.	
UNRECOGNIZABLE TABLE	EDITLIB
May be a bad binary or a PP program not allowed in a user library.	
UNSATISFIED EXTERNAL -- (name)	LOADO
UNSATISFIED EXTERNAL REF - (name)	LOAD
UNSATISFIED EXTERNALS	SEGBILD
VSNO ALREADY IN FAMILY	1DA
WAITING FOR RBT STORAGE	3DO,3PK,1DA
No empty chain member exists.	
WAITING FOR ASSIGNED UNIT xx	REQ
User requested specific EST ordinal which is currently assigned to another job. REQ waits for unit to become available.	
WAITING FOR OFF UNIT xx	REQ
User requested specific EST ordinal which is currently turned off. REQ waits for unit to be turned on.	
WAITING FOR FNT SPACE	REQ
WAITING FOR ROOM IN ECS	REQ
Requested ECS is not available. REQ waits for space to become available.	
WAITING FOR STORAGE	1BT,RST,RFL
WAITING FOR 15000 WORDS CM	1DF
WAITING FOR FNT SPACE	1DF

WAITING ALL QUIET	CKP
Checkpoint cannot be taken until all activity at the control point ceases.	
WAITING FOR nnnnnn CM	MEM
WAITING FOR ACCESS TO FILE	1PF
WAITING FOR PF UTILITY	1PF
WAITING FOR APF SPACE	1PF
WAITING FOR ARCHIVED FILE	1PF
WARNINGNEWPL CHECKSUM ERROR***	UPDATE
Dayfile message.	
WARNING, SEQUENCE NUMBERS ON NEXT CARD NOT IN ASCENDING ORDER	UPDATE
WPE UNRECOVERED. EOT FORCED, TYPE GO	XXXDMPQ
Permanent write error. Typing n.GO. will force EOT on bad reel and continuation reel will be assigned. Both reels must be read in when queue is restored.	
WRITE NOT AT EOI ON PERMANENT FILE	4ES
Fatal error, code 22.	
WRITE REQUIRES EXTEND PERMISSION	4ES,CIO
Fatal error, code 22.	
WRONG NUMBER OF PARAMETERS	COMBINE
XFER TABLE TOO LONG	LOADO
XFER ADDRESS NEGATIVE OR IN ECS	LOADO
XFER POINT UNKNOWN	LOADO
XREF LIST OVERFLOW - INCREASE THE SIZE OF BUF5	EDITLIB

YANK OR SELYANK IDENT OR YANKDECK NOT KNOWN UPDATE

Probably the ident referenced on a YANK or SELYANK card has been purged; applies to cards already on library. Non-fatal.

ZERO WORD COUNT IN A LOADER TABLE EDITLIB

1AJ CANT FIND LOADER 1AJ

Loader not in any system library.

1DL ABORTED DSD

DSD cannot find or load one of its overlays.

1GM CALLED AT IMPROPER TIME 1GM

Displayed if good morning routine called at other than 2400 hours.

1NR LOAD 1TF 1NR

1NR is loading CIO to load 1TF. Issued cnly when IP.DBUG=1.

1NR LOAD 1RP 1NR

1NR is loading 1RP. Issued only when IP.DBUG=1.

1NR RET USER 1NR

1NR is setting complete bit in the FET and dropping PP. Issued only when IP.DBUG=1.

1RS LOAD 1TF 1RS

1RS is loading 1TF. Issued only when IP.DBUG=1 .

1RS RET USER 1RS

1RS is setting the complete bit and dropping the PP. Issued only when IP.DBUG=1.

1RS LOAD 1RP 1RS

1RS is loading 1RP. Issued only when IP.DBUG=1.

4TH PARAMETER SHOULD BE B OR C SKIPF,SKIPB

Control card did not indicate binary or coded file.

63 FILES ALREADY ON PACK 5DA

668x MALFUNCTION 6WM

All type I/O drivers, except 1CT, 1CS, 1CR, 1C9 and 1RV, cause load 6WM to issue this message.

jobname STAGED BY OPERATOR 1MH

Issued after STAGE jobname type-in. (Release jobname to input queue).

jobname. ENTERED INPUT QUEUE 2TJ

8xx NOT IN LIB DSD

DSD could not find one of its routines in PP library.

uu BLANKED 1DA

Sent to system and control point dayfiles if pack with EST ordinal uu is blank labeled in response to a type-in.

x...x UNRECOGNIZABLE PARAMETER EDITLIB

x...x DIRECTIVE NAME - x...x UNKNOWN EDITLIB

x...x PARAMETER NAME - x...x UNKNOWN EDITLIB

x...x IS VALID ONLY DURING A CYBER 76 LIBEDIT RUN EDITLIB

x...x ALREADY IN x...x LIBRARY EDITLIB

x...x RESIDES ON DISK SINCE LIBRARY x...x NOT IN CM EDITLIB

x...x RESIDES IN ECS SINCE x...x LIBRARY NOT IN CM EDITLIB

x...x LIBRARY CONTAINS THE MAXIMUM NO. OF PROGRAMS EDITLIB

x...x IS NOT IN THE SOURCE LIBRARY - NO PROGRAMS ADDED EDITLIB

x...x IS NOT THE NAME OF THE NEXT RECORD ON FILE INPUT EDITLIB

x...x LIBRARY IS ALREADY IN THE LIBRARY NAME TABLE EDITLIB

x...x LIBRARY IS NOT A MEMBER OF THE LIBRARY NAME TABLE EDITLIB

x...x LIBRARY WILL RESIDE x...x EDITLIB

x...x LIBRARY NOT ADDED TO LIBRARY NAME TABLE EDITLIB

x...x LIBRARY IS EMPTY. LIBRARY WILL BE REMOVED EDITLIB

x...x ALREADY IN LIBRARY, NOT ADDED EDITLIB

x...x NOT ADDED EDITLIB

x...x IS NOT A LIBRARY EDITLIB

x...x IS DUPLICATE ENTRY POINT	EDITLIB
x...x NOT FOUND Program x...x did not appear in the interval specified.	EDITLIB
xxx NOT IN PPLIB Resident called to load overlay not named in library. Job terminated.	1EJ
xxx.x LIBRARY IS NOT INCLUDED IN THE x...x FILE	EDITLIB
xxxxxxx IS NOT A VALID DECK NAME Fatal error.	UPDATE
(x) DELIMITER IS MISSING	EDITLIB
/// A NAME APPEARS TWICE IN THIS STATEMENT	SEGBILD
/// A SYMBOL IN THIS STATEMENT HAS BEEN USED ON A DIFFERENT LEVEL	SEGBILD
/// A WORD FOLLOWED BY - IN THIS STATEMENT HAS ALREADY APPEARED AS A LABEL	SEGBILD
/// CARD EXHAUSTED BEFORE DIRECTIVE FOUND	SEGBILD
/// CLASHES WITH PREVIOUS GLOBAL/EQUAL STATEMENT	SEGBILD
/// COMMA NOT ENCLOSED BY PARENS	SEGBILD
/// DIRECTIVE UNRECOGNIZABLE	SEGBILD
/// FIRST SYMBOL IN SPEC. FIELD HAS ALREADY APPEARED AS LABEL	SEGBILD
/// LABEL ALREADY APPEARED AS TREE LABEL	SEGBILD
/// LABEL HAS APPEARED AS 1ST WORD IN SPEC. FIELD OF ANOTHER STATEMENT	SEGBILD
/// LEFT PAREN IN SPECIFICATION FIELD MUST BE PRECEDED BY -	SEGBILD
/// MORE THAN 1 TREE ON BOTTOM LEVEL	SEGBILD

/// NO TREE STATEMENTS ///	SEGBILD
/// SPECIFICATION FIELD STARTS WITH - ( ) OR .	SEGBILD
/// THIS LABEL HAS ALREADY APPEARED WITH FOLLOWING - IN ANOTHER STATEMENT	SEGBILD
/// THIS STATEMENT NO GOOD WITHOUT A LABEL	SEGBILD
/// THIS STATEMENT NO GOOD WITH EMPTY SPECIFICATION FIELD	SEGBILD
/// TOO MANY RIGHT PARENS SOMEWHERE	SEGBILD
/// WORD IN SPEC. FIELD HAS ALREADY APPEARED IN SPEC. FIELD OF ANOTHER STATEMENT OF SAME TYPE	SEGBILD
/// WOULD PRODUCE A CIRCULAR FAMILY TREE	SEGBILD
/// x...x IS INCLUDE OR GLOBAL STATEMENT LABEL, BUT NOT FOUND IN A TREE	SEGBILD
///// EMPTY PROGRAM RECORD IN LIBRARY /////	SEGBILD
///// 132K CM OVERFLOWED BY OBJECT PGMS /////	SEGBILD

TAPE AND DISK LABELS E

MAGNETIC TAPE LABELS

ANSI LABELS

SCOPE system file labels are defined for files recorded on 1/2 inch magnetic tape. These labels are designed to conform to the American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969. The SCOPE system provides for the processing of one additional label type, the format used by CDC 3000 Series Computer. Tapes containing a volume label (VOL1) or, optionally, a 3000 series header label are considered by the system to be labeled tapes. All other tapes are considered unlabeled.

SCOPE system labels and all 3000 series labels are recorded at the same density as the data on the tape. All system labels are 80 characters in length.

The following terms are defined in conjunction with SCOPE system tape label processing:

Volume — Synonymous with reel of magnetic tape.

File Set — One or more related files recorded on one or more volumes.

Tape mark — For 7-track tapes, a one-character record, 17 octal, plus one check character recorded in even parity. For 9-track NRZI tapes, a one-character record, 23 octal, plus one check character recorded in odd parity. For 9-track phase encoded tapes, 164 flux reversals at 3200 flux reversals per inch in tracks 1, 2, 4, 5, 7, 8 with tracks 3, 6, 9, DC erased. A tape mark indicates a boundary between files, between files and labels, or between certain label groups.

The first three characters of an ANSI label identify the label type. The fourth character indicates a number within the label type.

Type	No.	Label Name	Used At	
VOL	1	Volume header label	Beginning of volume	Required
UVL	1-9	User volume label	Beginning of volume	Optional
HDR	1	File header label	Beginning of file	Required
HDR	2-9	File header label	Beginning of file	Optional
UHL	†	User header label	Beginning of file	Optional
EOF	1	End-of-file label	End of file	Required
EOF	2-9	End-of-file label	End of file	Optional
EOV	1	End-of-volume label	End of volume	Required when appropriate
EOV	2-9	End-of-volume label	End of volume	Optional
UTL	†	User trailer label	End of file	Optional

†Any member of CDC 6-bit subset of ASCII character set.

All required labels are checked by the operating system on input and generated by the operating system on output if the user does not supply them. The user must supply all optional labels to the operating system.

The label processing mode is determined by the presence or absence of the XL bit in the FET for the file, bit 41 of word two of the FET.

TAPE FILE STRUCTURE

SCOPE standard system labels and tape marks establish the tape file structure according to the following rules. Required labels are indicated by a 4-character identifier, and tape marks are indicated by asterisks. In all the cases below, optional labels are placed as follows:

HDR2 through HDR9 may immediately follow HDR1.

EOF2 through EOF9 may immediately follow EOF1.

EOV2 through EOV9 may immediately follow EOV1.

UVL(a) (User Volume Label) may immediately follow VOL1.

UHL(a) (User Header Label) may immediately follow the last HDR(n) label.

UTL(a) (User Trailer Label) may immediately follow the last EOV(n) or EOF(n) label.

(a) indicates any 6-bit CDC character. (n) indicates any numeric character.

Single-Volume File

VOL1 HDR1* . . . Data Blocks . . . *EOF1**

Multi-Volume File

VOL1 HDR1* . . . First Volume Data . . . *EOV1**

VOL1 HDR1* . . . Last Volume Data . . . *EOF1**

Multi-File Volume

VOL1 HDR1* . . . FileA . . . *EOF1*HDR1* . . . . FileB . . . *EOF1**

Multi-Volume Multi-File

VOL1 HDR1* . . . FileA . . . *EOF1*HDR1* . . . FileB . . . *EOV1**

VOL1 HDR1* . . . Continuation of FileB . . . *EOV1**

VOL1 HDR1* . . . Last of FileB . . . *EOF1*HDR1* . . . FileC . . . *EOF1**

Volume Header Label

The first PRU in the volume must be a volume header label; it may not appear elsewhere.

File Header Label

Every file must be preceded by a file header label and every file header must be preceded by a tape mark or a volume header label. When a volume ends within a file, the continuation of that file in the next volume must also be preceded by a file header.

File Trailer Label

A file trailer label is required as the last block of every file. A file trailer must be preceded and followed by a tape mark, and if it is the last file trailer in the volume, two following tape marks are required.

Volume Trailer Label

When a volume ends within a file, the last PRU of the file in that volume must be followed by a volume trailer label which must be preceded and followed by tape marks.

When end-of-volume and end-of-file coincide, the labeling configuration is one of the following (* indicates tape mark):

```
VOL1 HDR1* . . . FileA . . . *EOV1**

VOL1 HDR1**EOF1*HDR1 . . . FileB . . . *EOF1**
     (A)   (A)  (B)

VOL1 HDR1* . . . FileA . . . *EOF1*HDR1**EOV1**
     (A)                           (B)

VOL1 HDR1* . . . FileB . . . *EOF1**
     (B)

VOL1 HDR1* . . . FileA . . . *EOF1*EOV1**
     (A)                      (A)

VOL1 HDR1* . . . FileB . . . *EOF1**
     (B)                      (B)
```

ANSI LABELS

The labels processed by SCOPE are the ANSI labels which follow:

Tape Label Formats – ANSI Standard Labels

	Character Position	Field	ANSI Name (SCOPE Name)	Length	Contents	Default Written	Checked On Input
Volume Header Label	1-3	1	Label Identifier	3	VOL	VOL	Yes
	4	2	Label Number	1	1	1	Yes
	5-10	3	Volume Serial Number	6	Any a characters	As typed from console	Yes if file assigned by VSN
	11	4	Accessibility	1	Space	Space	No
	12-31	5	Reserved	20	Spaces	Spaces	No
	32-37	6	Reserved	6	Spaces	Spaces	No
	38-51	7	Owner ID	14	Any a characters	Spaces	No
	52-79	8	Reserved	28	Spaces	Spaces	No
	80	9	Label Standard Level	1	1	1	No
First File Header Label	1-3	1	Label Identifier	3	HDR	HDR	Yes
	4	2	Label Number	1	1	1	Yes
	5-21	3	File Identifier (File Label Name)	17	Any a characters	Spaces	Yes
	22-27	4	Set Identification (Multi-File Set Name)	6	Any a characters	Volume Serial Number of first reel of set	No
	28-31	5	File Section Number (Reel Number)	4	4 n characters indicating number of volume in file	0001	Yes
	32-35	6	File Sequence Number (Position Number)	4	4 n characters indicating number of file in multi-file set	0001	Yes
	36-39	7	Generation Number	4	Not used by SCOPE	Spaces	No
	40-41	8	Generation Version Number (Edition Number)	2	2 n characters indicating the edition of file	00	Yes
	42-47	9	Creation Date	6	Space followed by 2 n characters for year, 3 n characters for day	Current date is used	Yes
	48-53	10	Expiration Date	6	Same as field 9	Same as field 9	Yes
	54	11	Accessibility	1	Any a characters	Space	No
	55-60	12	Block Count	6	Zeros	Zeros	Yes
	61-73	13	System Code	13	Any a characters	Spaces	No
	74-80	14	Reserved	7	Spaces	Spaces	No

Tape Label Formats — ANSI Standard Labels (Continued)

	Character Position	Field	ANSI Name (SCOPE Name)	Length	Contents	Default Written	Checked On Input
Additional File Header Labels	1-3	1	Label Identifier	3	HDR	HDR	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received by SCOPE from user.						
First End-of-File Label	1-3	1	Label Identifier	3	EOF	EOF	Yes
	4	2	Label Number	1	1	1	Yes
	5-54	3-11	Same as corresponding HDR1 label fields				
	55-60	12	Block Count	6	6 n characters; number of data blocks since last HDR label group		Yes
	61-80	13-14	Same as corresponding HDR1 label fields				
Additional End-of-File Labels	1-3	1	Label Identifier	3	EOF	EOF	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received by SCOPE from user.						
First End-of-Volume Label	1-3	1	Label Identifier	3	EOV	EOV	Yes
	4	2	Label Number	1	1	1	Yes
	All other fields are identical to EOF1 label.						
Additional End-of-Volume Labels	1-3	1	Label Identifier	3	EOF	EOF	Yes
	4	2	Label Number	1	2-9	2-9	Yes
	All other fields are not checked on input; they are written as received by SCOPE from user.						
USER Labels	1-3	1	Label Identifier	3	3 letter code: UVL, UHL, or UTL		Yes
	4-80	Any a characters. Content of these fields is not checked on input; is written as received by SCOPE from the user.					

3000 SERIES COMPUTER LABELS

Three Y labels are generated and processed by SCOPE: file header, EOF, and EOT.

Table E-2. 3000 Series Labels

	Character Position	Field	Name	Length	Contents
File Header Label	1	1	Density	1	Density at which label and data are recorded 2 = 200 bpi 5 = 556 bpi 8 = 800 bpi 8 = 1600 cpi
	2-3	2	Label Identifier	2	
	4-5	3	Logical Unit Number	2	Logical unit number; not checked by SCOPE
	6-8	4	Retention Cycle	3	3 n characters specifying number of days tape is to be protected from accidental destruction
	9-22	5	File Name	14	Any a characters to identify this file
	23-24	6	Reel Number	2	2 n characters to denote the particular reel in a series comprising the file
	25-30	7	Date	6	Date file was created; MMDDYY
	31-32	8	Edition	2	2 n characters distinguishing successive iterations of the same file
	33-80	9	User Information	48	Any a characters; SCOPE does not check this field
File Trailer Label	1-3	1	Label Identifier	3	EOF
	4-8	2	Block Count	5	5 n characters, number of data blocks written since file header label
	9-80	3	Unused	72	Unused
Volume Trailer Label	1-3	1	Label Identifier	3	EOT
	4-8	2	Block Count	5	5 n characters, number of data blocks written since file header label
	9-80	3	Unused	72	Unused

FAMILY DISK PACK LABEL FORMAT

The first two PRU's of a disk pack (64 60-bit words each) contain its basic label, formatted as follows. The words of each PRU are numbered 0-63; the bytes of each word are numbered 0-4, from left to right; the bits of a whole word or a byte are numbered 0-59 or 0-11, from right to left.

PRU 0

word 0	bits 36-59	DEV1
	bits 30-35	Binary zero
	bits 0-29	Julian date when label was written, yyddd in display code
word 1		Volume serial number of pack, 1 to 6 characters, right justified with 4 to 9 display code zeros as padding.
word 2		Binary zero
word 3		Binary zero for a blank-labeled pack. For a family pack:
	bits 18-59	Logical name of pack family, 1 to 7 characters left justified with binary zero padding.
	bits 12-17	Random bits
	bits 6-11	Family number of pack, binary, 0 to 4. Packs in a family are numbered as they are added in response to RPACK(pname,N) calls. Maximum number of packs in a family is 5.
	bits 0-5	Binary 0 unless family number is 0, in which case this field contains number of files in family, binary, 0 to 77 octal.
words 4-6		Binary zero
word 7	bytes 0-3	Binary zero
	byte 4	Checksum of other 319 bytes in PRU, formed as if by addition in a 12-bit accumulator with end-around carry.

words 8-9 — RBR header for pack; only significant fields are in word 9:

bits 54-59 — Half the length of RBR body, in CM words, as a binary number.

byte 4 — Total number of physically available record blocks in pack. (These two fields are not copied into RBR header in CMR when pack is attached to system; they serve as guides in formatting label.)

byte 3 — Number of logically available record blocks in pack, count of zero bits in body of RBR. This number is copied into corresponding position of RBR header in CMR when pack is attached to system. For a blank-labeled pack or pack not number 0 in family, this number is one less than the number in byte 4, indicating that only one record block is reserved for label. For pack number 0 in a pack family, it is n less than the number in byte 4, where n is the greatest number of record blocks that the label could occupy.

The RBR body begins at word 10 of PRU 0, and occupies the minimum number of words to furnish one bit for every physically available record block, according to the count in byte 4 of word 9. If this count is not exactly divisible by 60, the surplus bits at the end of the last word are set to 1, indicating non-existent record blocks. At the beginning of word 10, n bits are set to 1 to indicate record blocks reserved for the label and unavailable for file writing. n is the difference between the numbers in bytes 4 and 3 of word 9.

This RBR is copied into the proper RBR area in CMR when the pack is attached to the system.

If the RBR body is longer than 54 CM words, it extends into the beginning of PRU1. Otherwise, any words in PRU0 after its conclusion contain binary zero.

PRU 1

As explained above, if the RBR body is longer than 54 words, its 55th and following words are found in PRU 1. Apart from this, word 0-57 of PRU 1 contain binary zero.

For a blank-labeled pack, the remainder of PRU 1 contains binary zero, except that byte 4 of word 63 contains a checksum of all the other bytes in the PRU, formed by adding as if in a 12-bit accumulator with end-around carry.

In a family pack, words 58-62 contain a list of volume serial numbers of packs in its family. Pack number 0 (family serial number, not volume serial number) is noted in word 58, pack number 1 (if any), in word 59, and so on. The list is terminated by the first word, between words 60 and 63 inclusive, whose byte 0 contains binary zero. Everything after that zero byte, and before byte 4 of word 63, contains random bits. Byte 4 of word 63 contains the checksum of the PRU, as described in the preceding paragraph.

Each of the words containing the volume serial number of a pack in the family is formatted:

bits 54-59	Serial number of the pack in its family, a binary number between 0 and 4
bits 48-53	Random bits
bits 36-47	Number of files in the family if bits 54-59 contain 0; otherwise random bits
bits 0-35	Volume serial number of pack in the family, 1 to 6 characters right justified with 5 to 0 display code zeros as filler

PRU 2 AND FOLLOWING

For any pack that is not member number 0 of a pack family, the rest of record block 1 is ignored, and the space available for files begins at PRU 0 of record block 2. For member 0 of a pack family, the FNT entry model and the RBT chain for all files in the family are stored beginning with PRU 2 of record block 1, and on through as many record blocks as necessary. The total number of record blocks reserved for this purpose is indicated, as described above, in words 9 and 10 of PRU 0, record block 1.

The information for each file begins with a two-word FNT entry model as follows:

First word:	bits 18-59	File name, left justified with binary zero filler
	bits 12-17	Random bits
	bits 0-11	Binary zero
Second word:	bits 48-59	0700 octal
	bits 36-47	Binary zero if file is empty and has no RBR chain; otherwise non-zero but the exact value is not significant
	bits 0-35	Not significant

If the file is not empty, its RBT chain, consisting of word pairs, follows. These word pairs have the same form they had in the system RBT just before the pack family was detached from the system, and that they will have in the system RBT the next time it is attached to the system, except:

> Byte 0 of the first word of the last pair contains binary zero, to indicate the end of the chain for the file. Byte 0 of the first word of every other pair contains binary 1 for the first pair, 2 for the second, and so on.
>
> Bits 3-11 of byte 1 of the first word of each pair contains, not the RBR ordinal that was shown in the RBT when the pack was attached to the system, but the serial number, within the family, of the pack on which all the record blocks indicated in this word pair are to be found.

No special indication appears after the information for the last file of the pack family. The fact that it is the last file is determined from the file count in bits 0-5 of word 3, PRU 0, record block 1 of the same label.

SEQUENTIAL DISK PACK LABEL FORMAT

For a sequential pack, PRUs 0 and 1 of record block one are formatted as follows:

Word 0	DEV00yyddd; yy = year and ddd = day
Word 1	Visual identification of current pack; 1 to 6 letters and/or numbers right justified with 4 to 9 display code zeros as padding.
Word 2	0
Word 3	Visual identification of the first pack in a multi-pack set having an ID of $SEQzzzzzz where zzzzzz is 1 to 6 letters and/or numbers. If less than 6 characters are given, the visual identification is padded with display code zeros.
Word 4	0
Word 5	0
Word 6	Pack number (analogous to reel number on tape)
Word 7	Checksum of PRU 0
Words 8-9	RBR header–copied into CM when pack is assigned or when a new pack is referenced after having been mounted for end-of-pack condition and updated at job termination or when file or pack unit is closed.
Words 10-79	Body of RBR
Words 80-124	Zeros
Words 125-126	Zeros
Word 127	Checksum of PRU 1

RBT chains for the file begin with PRU 2 and continue as far as necessary.

PRINT FILE CONVENTIONS F

Files with a print disposition (including OUTPUT) and files assigned to a printer, must adhere to specific format rules as follows:

1. All characters must be in display code.
2. The end of a print line must be indicated by a zero byte in the lower 12 bits of the last central memory word of the line. Any other unused characters in the last word should be filled with display code blanks (55). For example, if the line has 137 characters (including carriage control), the last word would be aabbccddeeff550000 in octal; the letters represent the last seven characters to be printed in the line. No line should be longer than 137 characters.
3. Each line must start in the upper 6 bits of a CM word.
4. The first character of a line is the carriage control, which specifies spacing as shown in the following table. It will never be printed, and the second character in the line will appear in the first position. A maximum of 137 characters can be specified for a line, but 136 is the number of characters that will be printed. All characters apply to both the 501 and the 512 printer unless they are specifically designated otherwise.

CARRIAGE CONTROL CHARACTERS

Character	Action Before Printing	Action After Printing
A	Space 1	Skip to top of next page†
B	Space 1	Skip to last line of page†
C	Space 1	Skip to channel 6
D	Space 1	Skip to channel 5
E	Space 1	Skip to channel 4
F	Space 1	Skip to channel 3
G	Space 1	Skip to channel 2
H	Space 1	Skip to channel 1 (501) or Skip to channel 11 (512)
I	Space 1	Skip to channel 7 (512)
J	Space 1	Skip to channel 8 (512)
K	Space 1	Skip to channel 9 (512)
L	Space 1	Skip to channel 10 (512)
1	Skip to top of next page†	No space

† The top of a page is indicated by a punch in channel 8 of the carriage control tape for the 501 printer and channel 1 for the 512 printer. The bottom of page is channel 7 in the 501 and 12 in the 512.

Character	Action Before Printing	Action After Printing
2	Skip to last line on page	No space
3	Skip to channel 6	No space
4	Skip to channel 5	No space
5	Skip to channel 4	No space
6	Skip to channel 3	No space
7	Skip to channel 2	No space
8	Skip to channel 1 (501) or Skip to channel 11 (512)	No space
9	Skip to channel 7 (512)	No space
X	Skip to channel 8 (512)	No space
Y	Skip to channel 9 (512)	No space
Z	Skip to channel 10 (512)	No space
+	No space	No space
0(zero)	Space 2	No space
-(minus)	Space 3	No space
blank	Space 1	No space

When the following characters are used for carriage control, no printing takes place. The remainder of the line will be ignored.

Q	Clear auto page eject
R	Select auto page eject
S	Clear 8 vertical lines per inch (512)
T	Select 8 vertical lines per inch (512)
PM (col 1-2)	Output remainder of line (up to 30 characters) on the B display and the dayfile and wait for the JANUS typein /OKuu. For files assigned to a printer, n.GO. must be typed to allow the operator to change forms or carriage control tapes.
any other	Acts as a blank

Any pre-print skip operation of 1, 2, or 3 lines that follows a post skip operation will be reduced to 0, 1, or 2 lines.

The functions S and T should be given at the top of a page. In other positions S and T can cause spacing to be different from the stated spacing. Q and R need not be given at the top of a page as each will cause a page eject before performing its function.

GLOSSARY

Absolute Address

The actual physical location of a word in central memory. Contrast with relative address.

Allocatable Device

A storage device that can be shared by more than one job.

ATTACH

To make a permanent file accessible to a job by specifying the proper permanent file identification and passwords.

CATALOG

To place a file under jurisdiction of the permanent file manager routine, making it a permanent file.

Central Memory Resident (CMR)

Variable length low core area of central memory reserved for tables, pointers, and subroutines necessary for operation of the SCOPE system.

COMPASS

The assembly language of the CDC CYBER 70 and 6000 Series computers.

Control Points

The concept by which the multiprogramming capability of CDC CYBER 70 and 6000 Series computers is exploited. When a control point number is assigned to a job, that job is allocated some of the system resources; and it becomes eligible for assignment to the central processor for execution.

Dayfile

A chronological system permanent file, maintained on a permanent file device, which forms an accounting and job history file. Entries, called dayfile messages, are generated by operator action or by the system when control cards are processed or other significant action occurs. The system dayfile has entries for the entire system. Every job receives a job dayfile with entries pertinent to that job.

Deadstart

The process of initializing the system by loading the SCOPE system library programs and any of the product set from magnetic tape or a public device. Deadstart recovery is re-initialization after system failure.

Default

A system-supplied parameter value or name used when a value or name is not supplied by the user.

Dependency Count

A number established by the user with the Dyn parameter on a job card and decremented by other jobs in the dependency string. The job will not be run until the count reaches zero.

Dependent Job

A job which depends on the execution of other jobs before it can be run. It cannot be run until its dependency count is zero.

Device Type Code (dt)

An optional parameter on a REQUEST card which specifies the type of device on which the named file is to be stored. It may encompass a group of parameters to define the device characteristics in detail.

Display Code

Character code used internally in the computer. Each character consists of 6 bits (2 octal digits).

DISPOSE

The DISPOSE card or macro specifies the device, site, and form on which the named output file is to be processed; also it may specify whether processing is to occur immediately or at end of job.

Disposition Code

A two-character mnemonic indicating device, site, form, and format for processing a file named on a DISPOSE card or macro. Also, an octal value returned to the file environment table corresponding to the ultimate disposition of the file.

ECS

Extended core storage

EDITLIB

A utility program which allows creation or maintenance of library files suitable for use by the loader.

End-of-File

Indicates end of a file. In card decks, a card with 6/7/8/9 multiple punch in column 1, or a 7/8/9 multiple punch in column 1 and an octal level number of 17 in columns 2 and 3. On mass storage devices and SCOPE tapes, a marker with an octal level number of 17 in a zero length physical record unit. On S and L tapes, a single tape mark.

End-of-Information

Physical end of data. Equivalent to end-of-file in a card deck. On SCOPE tapes and on labeled S and L tapes, an EOF trailer label followed by two tape marks. On mass storage devices, the position of the last written data.

End-of-Record (EOR)

Indicator of end of a SCOPE logical record. In card decks, a card with 7/8/9 multiple punch in column 1 and optionally an octal level number 0-16 in columns 2 and 3. On mass storage device or SCOPE tapes, a marker with an octal level number 0-16 contained in a short or zero length physical record unit.

End-of-Tape Reflective Marker

A reflective strip near the end of a magnetic tape. It is used to signal termination of operations on a particular volume. At least 18 feet of tape must follow this marker.

EST Ordinal

The number designating the position of an entry within the equipment status table established at each installation. Every hardware device is identified by its ordinal.

Family Pack

A non-allocatable disk pack which can hold one or more sequential or random files. As many as five packs can exist in the family to hold up to 63 files; but all packs in a family must be mounted at once.

Field Length (FL and FE)

FL is the number of central memory words a job requires. FE is the number of words in the direct access area of extended core storage that a job requires. Within central memory or extended core storage, the field length added to the reference address defines the upper address limit of a job.

File Environment Table (FET)

A table used for communication between a user program and the operating system when files are processed. An FET created by a compiler or by the assembly language programmer is required within the user field length for each file in the program.

File Set

One or more related files recorded on one or more volumes.

Hang

A system stop that may be caused by hardware failure or by an error in a SCOPE system program. An error in a user program could cause that program to hang (go into a loop or abort), but no user program error should cause a system hang.

INPUT

A logical file name assigned by SCOPE to every job. It contains the user job deck.

JANUS

A group of SCOPE peripheral processor routines which control the processing of up to 4 card readers, 3 card punches, and 12 line printers. It normally functions at control point 1, but may be assigned to another control point by the operator.

L Tape

A 7-track labeled or unlabeled magnetic tape containing physical records whose sizes range from one central memory word to an upper limit specified by the size of the buffer for that tape.

Labeled Tape

A magnetic tape with header and trailer labels having the format of the CDC CYBER 70 or 6000 Series standard labels or the 3000 Series labels; Alternately a tape with non-standard labels.

Level

An indicator specifying relative position in a hierarchy. For priority considerations, level 0 is the lowest priority. For SCOPE logical records, octal level numbers 0-17 can be used to organize files. For overlay and segment loading, a pair of numbers specifies level, with (0,0) being the portion of the program remaining in memory.

Level Number

An octal number from 0-17 in a short physical record unit or zero-length physical record unit marker to form SCOPE logical record groups within files. Level number 17 indicates a logical end-of-file. Level number 16 is used by checkpoint/restart and should not otherwise be specified by the user. SCOPE creates logical records with a level number of 0 for mass storage files and SCOPE tapes when the user does not specify otherwise.

Library

A file or collection of files containing executable programs and Tables needed to locate and load the programs. System libraries can contain peripherial processor programs in addition to the central processor programs.

Load Point

The reflective marker near the beginning of a magnetic tape. Data, including labels, is written after the load point. A rewind positions a single file reel to the load point. At least 10 feet of tape should precede the load point marker.

Logical File Name (lfn)

The 1-7 display coded alphabetic or numeric characters by which the operating system recognizes a file. Every lfn in a job must be unique and begin with a letter.

Macro

A COMPASS language statement which generates other source language code.

Monitor

The SCOPE routine which coordinates and controls all activities of the computer system. It occupies peripheral processor 0 and part of central memory. It schedules the use of the central processor and the other peripheral processors.

Non-allocatable Device

A device such as a magnetic tape or a family pack which can be used by only one job at a given time.

NUCLEUS

A system library containing the essential SCOPE programs needed to load and execute all other system library programs. It is available to all jobs without explicit call.

OUTPUT

A logical file name assigned by SCOPE to each job to receive information such as assembly listing, diagnostics, load map, dayfile, and program output. It is printed at job termination unless otherwise disposed by the user.

Password

A word on a permanent file CATALOG establishing a privacy control that must be duplicated on an ATTACH to have file access.

Permanent File

A mass storage file cataloged by the system so that its location and identification are always known to the system. Permanent files cannot be destroyed accidentally during normal system operation (including deadstart), and they are protected by the system from unauthorized access according to privacy controls specified when they are created.

Physical Record Unit (PRU)

The smallest amount of information transmitted by a single physical operation of a specified equipment, measured in central memory words. A PRU for mass storage devices is 64 decimal words long; that for SCOPE binary magnetic tape is 512 decimal words; etc.

Private Device

A non-allocatable mass storage device which can be used only by specific request. It is logically removable. Family packs and sequential packs are private devices.

Public Device

An allocatable mass storage device available to SCOPE for assignment of default residence files.

PUNCH

A logical file name that causes the file to be punched on cards in Hollerith format when the job terminates.

PUNCHB

A logical file name which causes the file to be punched on cards in binary format when the job terminates.

Random File

A file with an index to each record in the file.

Recall

The state of a program when it has released control of the central processor until a fixed time has elapsed or until a requested function is complete. Recall is a system action request, as well as an optional parameter of some file action requests.

Record Manager

A software product, running under SCOPE, which allows a variety of record, blocking, and file types to be created and read.

Reference Address (RA and RE)

RA is the absolute central memory address that is the starting, or zero relative address for a program. Addresses within the program are relative to RA. RA + 1 is used as the communication word between the user program and Monitor. RE is the absolute extended core storage starting address used by a program.

Relative Address

All addresses in a relocatable program are relative to a base address of zero. When a relocatable program is loaded for execution, the zero base address is assigned to a reference address. At that time, all addresses in the program become relative to the reference address.

Rotating Mass Storage (RMS)

Disk, disk pack, or drum storage device.

S Tape (stranger tape)

A magnetic tape (labeled or unlabeled, 7 or 9 track) containing physical records ranging in size from 2 characters to 5120 decimal characters. This tape does not contain any level numbers.

Scheduler

A group of SCOPE routines which select jobs for assignment to control points and control swapping and rollout of jobs.

SCOPE

The operating system for CDC CYBER 70 and 6000 Series computers. The name is derived from Supervisory Control of Program Execution.

SCOPE Logical Record

A data grouping that consists of one or more physical record units immediately followed by a short physical record unit or a zero-length physical record unit. These records may be transferred between devices without loss of data or structure.

SCOPE Tape

A tape created under SCOPE with fixed length physical record units: for coded tape, 128 decimal central memory words; for binary tape, 512 decimal central memory words. A SCOPE tape may be labeled or unlabeled, and written on 7- or 9-track tape.

Sequential File

A file in which records must be located by position, not address.

Sequential Pack

A private pack which can hold a single sequential file. The file may extend to any number of packs, only one of which need be mounted for processing at a given time.

Short PRU

A physical record unit containing data and a marker with an octal level number to mark the end of a SCOPE logical record; length of this information is less than the standard PRU size of the storage device.

Staging

Releasing a tape job from the tape queue for scheduling.

Standard Labeled Tape

A tape with labels conforming to American National Standard Magnetic Tape Labels for Information Interchange X3.27-1969. Also called a system labeled tape.

Swapping

The concept of removing jobs from central memory to mass storage before execution is complete, so control point and memory can be assigned to another job. A job is swapped out when it is waiting for an external event, or when its control point and/or central memory is needed by a higher priority job.

System Libraries

The collection of tables and object language programs residing in central memory or on mass storage which are necessary for running the SCOPE system and its product set.

Tape Mark

A short record written on tapes under SCOPE control to separate label groups, files, and/or labels. Interpretation depends on the tape format. (See page E-1.)

Unlabeled Tape

A magnetic tape that does not have a header label. Unlabeled tapes generated by SCOPE contain a trailer label similar to the trailer for a standard labeled tape.

Unsatisfied External

An external reference in a user program for which the system does not have a link to a user program or a system library program after the user program is loaded.

UPDATE

A utility program that allows a source statement program stored on mass storage or tape to be modified and restored.

User Library

Library file a programmer created through the EDITLIB program containing loader tables referencing the assembled central processor programs, subroutines, text records, or overlays.

Volume

A term synonymous with reel of tape.

Zero Length PRU

A physical record unit, containing only an octal level number, that is used to terminate a SCOPE logical record; it does not contain any data.

INDEX

COMMENT SHEET

CONTROL DATA
CORPORATION

TITLE: SCOPE Reference Manual Version 3.4

PUBLICATION NO. 60307200 REVISION D

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

General comments:

FROM NAME: __________ POSITION: __________

COMPANY
NAME: __________

ADDRESS: __________

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

CUT ON THIS LINE

STAPLE STAPLE

.D FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Documentation Department
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

.D FOLD

STAPLE STAPLE